




# Lab 9 – Design of an Instruction-Fetch Datapath

## Objectives

In this lab, we will develop an instruction-fetch datapath and verify its functionality.

Section	
a) <u>Introduction</u>  A brief overview of this lab.	10
b) <u>Implementation</u>  In this section, you will implement a single-cycle RISC processor for R-type instructions and simulate its behavior.	80





## a. Introduction

A reasonable way to start a datapath design is to examine the major components required to execute each class of RISC-V instructions. First, an instruction has to be fetched, which requires the components shown in the following figure.

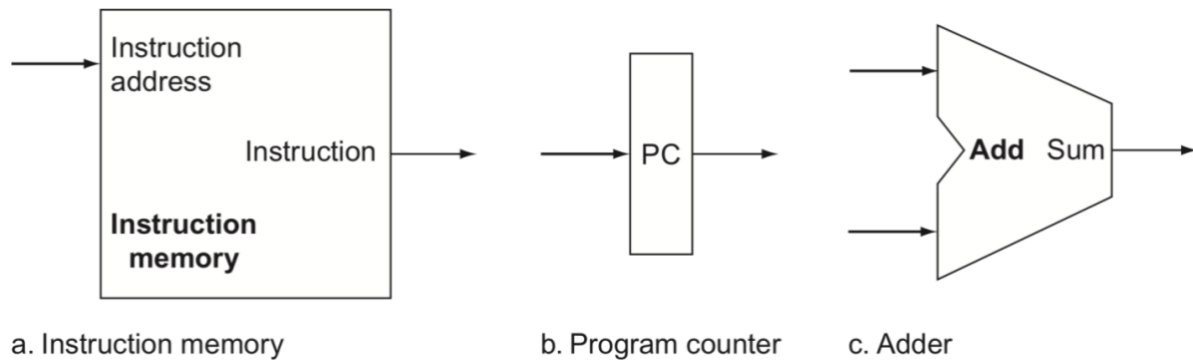


Fig. 9.1. Required components for fetching a processor instruction.

As shown in Fig. 9.1., two state elements are needed to store and access instructions, and an adder is needed to compute the next instruction address. The state elements are the instruction memory and the program counter. The instruction memory need only provide read access because the datapath does not write instructions. Since the instruction memory only reads, we treat it as combinational logic: the output at any time reflects the contents of the location specified by the address input, and no read control signal is needed. The program counter is a 64-bit register that is written at the end of every clock cycle and thus does not need a write control signal. The adder is an ALU wired to always add its two 64-bit inputs and place the sum on its output.

## a. Implementation

We already have developed an instruction memory in Lab08. Now, we will start off with developing the separate modules of a program counter (PC) and a two-input adder.

### i. Lab Task 01

Write a module, named `Program_Counter`, which takes three inputs – `clk`, `reset`, a 64-bit input, `PC_In`; and a 64-bit output, `PC_Out`. Initialize `PC_Out` to 0 if `reset` signal is high, else reflect the value of `PC_In` to `PC_Out`, at the positive edge of clock.

### ii. Lab Task 02

Adder takes two 64-bits inputs, `a` and `b`; add them, and reflect the results at the 64-bit output port, `out`.





### iii. Lab Task 03

Now connect the above two modules and an `Instruction_Memory` (developed in Lab08) to construct an instruction fetch datapath as shown below. Name the module `Instruction_Fetch`; instantiate all three modules and make necessary connections as shown in Fig. 9.2.

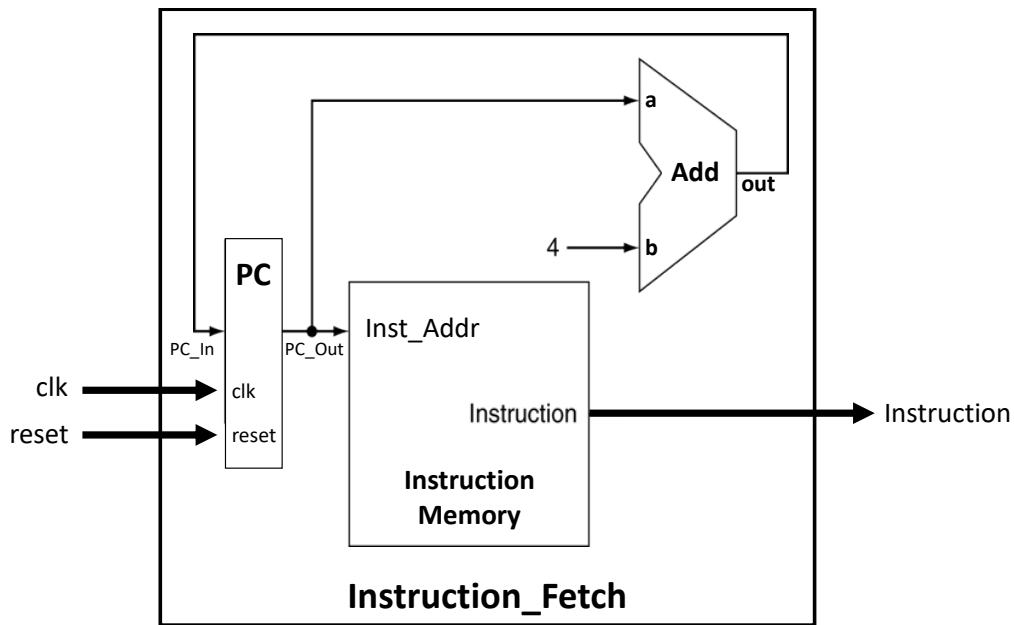


Fig. 9.2. Instruction fetch datapath.

Write a testbench and verify the functionality of instruction fetch datapath.

**Assessment Rubric**  
**Computer Architecture Lab**  
**Lab 09**

**Design of an Instruction-Fetch Data path**

<b>Name:</b>	<b>Student ID:</b>
--------------	--------------------

**Points Distribution**

<b>Task No.</b>	<b>LR 2 Code</b>	<b>LR 5 Results</b>	<b>AR 7 Report Submission</b>
<b>Task 1 Program Counter</b>	/20	-	/20
<b>Task 2 Adder</b>	/15	-	
<b>Task 3 Instruction Fetch</b>	/25	/20	
<b>Total Points</b>	/100 Points		
<b>CLO Mapped</b>	CLO 1		

*For description of different levels of the mapped rubrics, please refer to the provided Lab Evaluation Assessment Rubrics.*

#	Assessment Elements	Level 1: Unsatisfactory Points 0-1	Level 2: Developing Points 2	Level 3: Good Points 3	Level 4: Exemplary Points 4
LR2	<b>Program/Code/ Simulation Model/ Network Model</b>	Program/code/simulation model/network model does not implement the required functionality and has several errors. The student is not able to utilize even the basic tools of the software.	Program/code/simulation model/network model has some errors and does not produce completely accurate results. Student has limited command on the basic tools of the software.	Program/code/simulation model/network model gives correct output but not efficiently implemented or implemented by computationally complex routine.	Program/code/simulation /network model is efficiently implemented and gives correct output. Student has full command on the basic tools of the software.
LR5	<b>Results &amp; Plots</b>	Figures/ graphs / tables are not developed or are poorly constructed with erroneous results. Titles, captions, units are not mentioned. Data is presented in an obscure manner.	Figures, graphs and tables are drawn but contain errors. Titles, captions, units are not accurate. Data presentation is not too clear.	All figures, graphs, tables are correctly drawn but contain minor errors or some of the details are missing.	Figures / graphs / tables are correctly drawn and appropriate titles/captions and proper units are mentioned. Data presentation is systematic.
AR7	<b>Report Content/Code Comments</b>	Most of the questions are not answered / figures are not labelled/ titles are not mentioned / units are not mentioned. No comments are present in the code.	Some of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Few comments are stated in the code.	Majority of the questions are answered, figures are labelled, titles are mentioned and units are mentioned. Comments are stated in the code.	All the questions are answered, figures are labelled, titles are mentioned and units are properly mentioned. Proper comments are stated in the code.