CS311: Computer Architecture Lab

Assignment 5 - Report

Anudeep Tubati 170010039
S V Praveen 170010025

The aim was to develop a pipelined processor Simulator for ToyRISC.

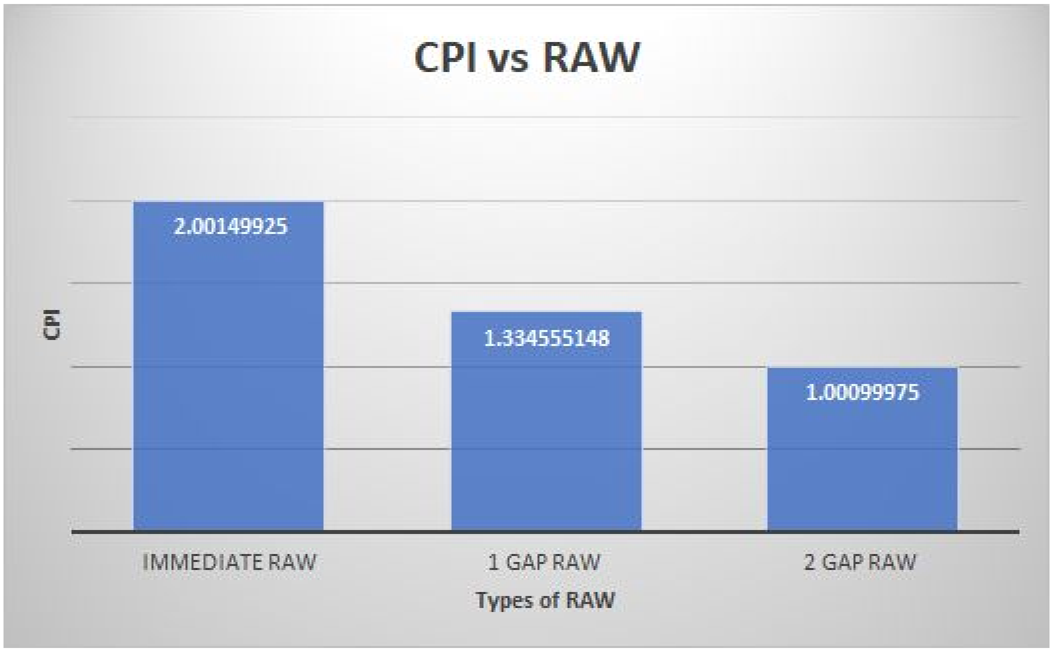
Statistics -

Sl No.	***Program***	***No. of Instructions executed***	***No. of cycles taken***	***No. of OF Stalls***	***No. of Wrong Instructions***
1.	*descending.asm*	*277*	*734*	*162*	*176*
2.	*even-odd.asm*	*6*	*18*	*5*	*0*
3.	*fibonacci.asm*	*78*	*153*	*36*	*32*
4.	*palindrome.asm*	*49*	*117*	*40*	*14*
5.	*prime.asm*	*29*	*87*	*19*	*10*

Table: *Statistics of Processor performance on Test Programs*
(provided in the assignment)

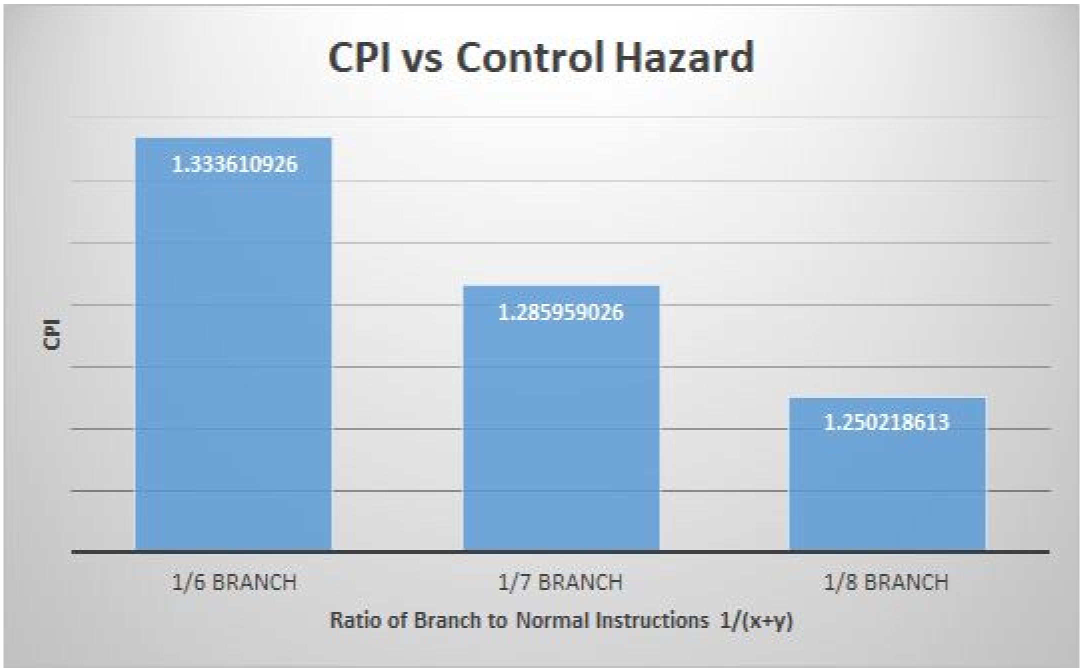
Observations

1) **Data Interlocks**
   - For programs that have **no gaps** in between RAW hazards, the expected **CPI is 2** (since RW->OF forwarding happens without being explicitly programmed, the expected 3 CPI is brought down to 2. This forwarding occurs as the 5 stages aren't exactly parallel in actual code execution).
   - For programs which have **one gap** in between RAW hazards, the expected **CPI is 1.33**
   - For programs which have **two or more gaps** in between RAW hazards, the expected **CPI is 1**



2) Control Interlocks

- Generally, for every branch in a set of instructions, 2 cycles are extra as the branch is detected in the EX stage. Hence the general formula for CPI when there are x branches in y instructions (assuming no RAW hazards) is (3x + y)/(x +y)
- The graph shown below has been constructed by using x = 1 and y = 5, 6, 7 and it clearly aligns with the formula obtained before.


CPI vs Control Hazard
CPI
1.333610926
1.285959026
1.250218613
1/6 BRANCH
1/7 BRANCH
1/8 BRANCH
Ratio of Branch to Normal Instructions 1/(x+y)