CS311 (Computer Architecture Lab) **Assignment-04**

VIDIT JIGNESH PARIKH

March 11, 2025

1 Introduction

In this assignment, we simulated the working of a pipelined processor model using Java. Each stage of processing (and the latches in between) are coded as Java classes. The instructions queue into the processor, and if a conflict is detected, then the instructions in the previous latches are discarded. While this ensures correctness, it is at the cost of time and efficiency. We would work on improving this issue in the upcoming assignments and by the end of 6th assignment, we will have a fully functional processor, made from scratch made over the semester!

2 Results - Tabulation

Object File Name	**# Cycles**	**# `OF` stalls**	**# wrong `BRANCH` instructions**
`descending.out`	658	126	220
`evenorodd.out`	16	8	4
`fibonacci.out`	157	44	36
`prime.out`	75	15	28
`palindrome.out`	116	43	18

Table 1: Results Table

3 Observations

Let us recall the data from the non-pipelined processor from the previous assignment:

Object File Name	**# Cycles**
`descending.out`	365
`evenorodd.out`	6
`fibonacci.out`	94
`prime.out`	34
`palindrome.out`	56

Table 2: Assignment-3 Observations Compilation

One can observe that the number of cycles in a pipelined processor are almost $\frac{1}{2}$ as many of those which we found in for a non-pipelined processor. Theoretically, we expect this value to be $\frac{1}{5}^{th}$ the non-pipelined number of cycles. This disrepancy can be assigned to the various kinds of data hazards that are faced while execution (and hence the implementation of `NOPs` which stall the cycle and hence increase their number).

Additionally, the number of *wrong* branch instructions and the number of `OF (Operand Fetch)` stages stalls are affected depending upon the number of data hazards came across while the execution of programme takes place.