# CS 311, Assignment 4 - Report

Thummala Uttam Kumar Reddy (200010052),

Bharath Chandra Panasa (200010038)

October 07, 2022

## Abstract

In this assignment, we updated our *simulate* function to a loop implements following algorithm:

```
while (not end of simulation) {
    performRW performMA
    performEX performOF performIF
    increment clock by 1
}
```

In this stage, we worked on the output of the previous stage in the previous cycle. If the input latches to an invalid stage, the stage must do nothing and proceed.

## Statistics Table

The table below (Table 1) details the number of cycles consumed by each benchmark programme, the number of times the OF stages had to halt due to a data hazard, and the number of times an instruction on the incorrect branch path entered the pipeline.

## Interpretation and Comments

We can observe that the number of cycles required by each programme with pipeline implementation is reduced to half that of the programme without the pipeline. In an ideal world, it could be reduced to one-fifth of the total number of cycles. In the event of data hazards, we add NOP instructions to avoid hazards, which increases the overall number of cycles.

Similarly, the amount of incorrect branch instructions and OF instruction stalls are affected for the same reasons mentioned above in order to avoid dangers.

| File Name | No.of Cycles | No. of Wrong Branch Instructions | No. of OF Stalls |
|---|---|---|---|
| descending.out | 658 | 220 | 126 |
| evenorodd.out | 19 | 4 | 10 |
| fibonacci.out | 157 | 36 | 44 |
| palindrome.out | 124 | 18 | 51 |
| prime.out | 79 | 28 | 19 |