# Computer Architecture Assignment 4

Aryan Gulhane 210010006
Srihari K G 210030035

October 8, 2023

1. **Abstract -**

   Abstract: In this assignment, we made significant improvements to our simulation function by implementing a loop structure. This loop operates as follows:

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

   Each stage processes the output generated by the preceding stage in the previous cycle. If the input data for a stage is invalid (for example, during the very first cycle when the MA-RW latch has no valid content), the stage takes no action.

2. **Statistic Table**

   Table 1 below provides information on the number of cycles required for each benchmark program, the number of static instructions , number of branches locked due to branch instructions , number of data conflicts raised and the number of instructions fetched in IF stage.

| ToyRISC Program File | Number of Static Instructions | Number of Cycles | Number of Branches locked | Number of Data Clashes | Number of Instructions Fetched |
|---|---|---|---|---|---|
| Prime.asm | 16 | 65 | 5 | 12 | 36 |
| Palindrome.asm | 16 | 133 | 7 | 33 | 56 |
| fibonacci.asm | 21 | 158 | 16 | 23 | 88 |
| evenorodd.asm | 9 | 18 | 0 | 5 | 8 |
| descending.asm | 29 | 637 | 88 | 91 | 352 |

3. **Interpretation and comments**

   Our observations reveal a significant reduction in the number of cycles required for each program when implementing the pipeline. Ideally it could reduce the number of cycles to one fifth of without pipeline implementation.

   The data interlocks is implemented by firstly detecting the clashes in the latches and then removing the instructions in OF and IF and introducing the instruction in OF to IF later.

   Whenever branch instructions are encountered, the instructions fetched after it will be destroyed and only when it is taking a branch the new instruction is fetched otherwise the next instruction is fetched after the halting.

   We may note that in EvenOdd code we encountered no branch are locked and we see more number of cycles in descending code as there were more number of data and branch interlocks involved.