CS311
Assignment-04

Chidaksh Ravaru (200010046)
Pavan Kumar V Patil (200030041)

October 7, 2022

1 Abstract

In this assignment-5 we have to update simulate function to a loop that looks something like pseudocode given below:

```
while( not end of simulation )
{
    perform_RW
    perform_MA
    perform_EX
    perform_OF
    perform_IF
    increment clock by 1
}
```

Each stage must operate on the output generated by the previous stage in the previous cycle. If the input latch to the stage has invalid content, the stage must do nothing means no operation(nop). Depending upon which hazard (data/control) we have to decide how many cycles to stall.

2 Statistics

Programs	No. of Cycles	No. of OF stage stalls	No. of wrong branch instructions
descending.out	543	210	52
evenorodd.out	14	4	0
fibonacci.out	130	40	8
palindrome.out	93	33	7
prime.out	53	19	1

Table 1: Statistics Table

3 Comments on observation

We get to know that for large number of dynamic instructions $IPC \approx 1$. For the first instruction to come out of pipeline it takes 5 cycles, for second it takes 6 cycles, for third it takes 7 cycles, and so on till program ends, so we can say that for n instructions it takes n+5 cycles to run, therefore $IPC \approx 1$, this is the case for ideal pipeline. For non-ideal pipeline (i.e which has data and control hazards) the performance is little low compared to ideal pipeline because if data hazard occurs we have to stall IF and OF stage for 3/2/1 cycles according to instructions and for control hazard we have stall for 2 cycles, due to which IPC decreases, so performance of processor decreases.

- descending.out: This program is quite long as it has many loops due to which dynamic instructions of the program increases. That's why it takes 543 cycles to complete its execution. It contains many data dependencies so processor has to stall IF and OF many times. Due to loops taken again and again it increases control hazards.
- evenorodd.out: This program is short and given input is '11' in benchmark program. There is only one branching in this program which is never taken as input is odd, so control hazard never occurs. There are 4 stalls due to data dependencies 2 times (between load and divi and between sub and addi).
- fibonacci.out: Program has many loops, due to which it has many dynamic instructions to run, therefore it has data hazards and control hazards.
- palindrome.out: similar to above, program has many loops, due to which it has many dynamic instructions to run, therefore it has data hazards and control hazards.
- prime.out: Program has only 2 branches which will be taken when deciding to be done. This deciding will come only once at the end due to which one out of two branch paths is definitely is going to be correct due to which control hazard occurs.