CS311 - Computer Architecture Lab

Assignment 5 - Report

- Sourabh Bhosale (200010004)
- Surzith Raj Pandi (200010049)

1. Statistics Table:

- Total no. of cycles taken by each benchmark program.
- Total no. of instructions executed.
- The throughput in terms of instructions per cycle

All the calculations for the lab assignment are tabulated below.

	Total No. of Cycles	Total No. of instructions	Throughput
descending	11433	365	0.03192513
evenorodd	249	6	0.024096385
fibonacci	3157	94	0.029775104
palindrome	1983	56	0.02824004
prime	1175	34	0.02893617

2. Interpretation and Comments:

For odd-even, code has no control hazards because the conditional branch was not taken as the value checked was odd. There are a few RAW hazards due to which we see a small number of data hazards.

For fibonacci, there is a control hazard in the loop. Every time it goes in the loop, it is a wrong branch, till the count reaches 10. So, we see a few control hazards. Similarly, there are a few data hazards outside the loop and a few inside. Hence, we also see some stalls in the pipeline.

For descending, code has a lot of branching, 3 nested loops and data hazards. Hence, we see that the total number of hazards increases by a large amount and so does the cycles executed.

For Prime, the loop runs from n=2 to n=5, i.e. 4 times. All these 4 times, we get a wrong branch taken for the jmp statements. Then it takes a wrong branch when it is supposed to enter prime/not prime. Hence, a total of 10. Similarly, we are getting a few data hazards.

We may conclude that that every time the code misses the jump to loop, there is a wrong branch taken, thus we get a control hazard. But we have more data hazards per loop. Hence, we see a larger number of data hazards than control hazards. We additionally add NOP instructions to avoid hazards and reorder the code. Here in all cases we can commonly observe that throughput is almost the same as it is basically a proportion of instruction per cycle. Here, it is in between between 0.024 and 0.032.