Computer Architecture Lab 5

PRIYANSHU RAO

April 2025

File Name	Number of Cycles Taken	Number of Instructions	IPC
PRIME	1573	29	0.01843611
DESCENDING	20806	277	0.013313468
EVENORODD	249	6	0.024096385
FIBONACCI	4465	78	0.017469205
PALINDROME	2581	49	0.01898489

Table 1: Performance Metrics of Test cases Files

File Name	Number of Cycles Taken	Number of Instructions	IPC
PRIME	2939	51	0.017352842
DESCENDING	22029	348	0.015797358
EVENORODD	329	6	0.018237082
FIBONACCI	4259	85	0.019957736
PALINDROME	775	14	0.018064516

Table 2: Performance Metrics of Benchmark Files

Observations and Analysis on Test cases Assembly files

- **Number of Instructions:**
  Simple programs like *evenorodd.asm* and *palindrome.asm* have less number of instructions due to their simple logic. While programs like *descending.asm* and *fibonacci.asm* have a higher number of instructions because of loops, iterations, and more complex operations.

- **Number of Cycles:**
  The number of cycles taken is highest for the *descending.asm* benchmark,

indicating that it has repetitive operations like loops for sorting tasks. The *evenorodd.asm* program has the least number of cycles as it involves simple conditional checks for even or odd.

- **IPC (Instructions Per Cycle):**
  IPC values are generally low for all programs, which indicates limited parallel execution. Programs like *evenorodd.asm* show a higher IPC value due to their simple execution with fewer branches. On the other hand, *descending.asm* has the lowest IPC due to frequent branching, memory access, and conditional statements that reduce pipeline efficiency.

Overall, the benchmarks with simpler logic and fewer branches (like *evenorodd.asm*) demonstrated higher IPC and lower cycle counts. Complex programs (like *descending.asm* and *fibonacci.asm*) with loops and conditions have higher number of instructions and cycles and lower IPC