CS311: Lab Report
Assignment 5 - Discrete Event Simulator

B Siddharth Prabhu
`200010003@iitdh.ac.in`

Devdatt N
`200010012@iitdh.ac.in`

20 October 2022

1 Introduction

In this assignment, we simulated the working of a discrete event model using Java. An event is a tuple of the form: < `event_time` , `event_type` , `requesting_element` , `processing_element` , `payload` >. The event queue is a list of events ordered by time. An event is said to "fire" when the current clock cycle is equal to the event time. When this happens, the `handleEvent()` function of the processing element is invoked. Handling of an event may in turn lead to more events being generated, for the same clock cycle, or for some future clock cycle. Main Memory Latency is modeled, and this is reflected in both the IF and MA stages.

2 Input and Output of the Program

Inputs include:

1. Full path to configuration file, `src/configuration/config.xml` .
2. Full path to statistics file, `stats.txt` , which stores statistics of the simulation run.
3. Full path to object file, for example `text_cases/descending.out` whose execution is to be simulated.

We run the program for a given object file (e.g. descending.out) using the following command-line argument(s), which have been put into a shell script for simplicity:

```
#!/bin/bash
ant;
ant make-jar;
java -jar jars/simulator.jar src/configuration/config.xml stats.txt
↪  test_cases/descending.out;
```

Output includes the statistics file which must be created at the required location. (`ant;` and `ant make-jar;` do not have to be used in runs after the first simulation.) Pre-existing jar/bin folders may raise minor issues, which can be overcome by simply removing them (since they are generated again anyway).

3 Scope for Future Improvement

We have tabulated statistics for all given programs in the next section. We can improve on this using hardware methods like introduction of caches.

4 Tabulation of Observations

Benchmark Object File	Number of Cycles	Number of Instructions	CPI	Throughput (in terms of IPC)
`descending.out`	13196	329	40.10942249	0.024931798
`evenorodd.out`	246	6	41.0	0.243902439
`fibonacci.out`	3464	86	40.27906977	0.024826789
`prime.out`	1218	30	40.6	0.024630542
`palindrome.out`	2274	56	40.60714286	0.024626209

Table 1: Observation Table

5 Observations

- Main Memory latency has been modeled. It is reflected in IF stage and during load/store processes.
- Latencies of different functional units have been modeled. This is reflected in EX stage, and includes components such as ALU, multiplier, divider, etc.
- Main memory latency is observed to be roughly 40 for our processor configuration.
- If programs have more data and control hazards, then it will have more cycles per instruction.