CS311: Lab Report
Assignment 6 - L1 Cache Simulation

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

Devdatt N
`200010012@iitdh.ac.in`

4 November 2022

1 Introduction

In this assignment, we added caches to the simulated memory system.

- One cache (instCache) was added between the IF stage and the main memory. Let us call this the "level 1 instruction cache" or the `L1i-cache` .
- One cache (dataCache) was added between the MA stage and the main memory. Let us call this the "level 1 data cache" or the `L1d-cache` .

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 Cache Configurations

Cache size	16B	128B	512B	1kB
Latency	1 cycle	2 cycles	3 cycles	4 cycles
Line Size	4B			
Associativity	2			
Write Policy	Write Through			

Table 1: Cache Configurations

4 Analysis and Tabulation

4.1 Varying Instruction Cache

First we fix the size of the `L1d-cache` at 1kB. Then, we vary the size of the `L1i-cache` from 16B to 1kB (remember to change latency accordingly), and study the performance (instructions per cycle). Plots are included later.

Assembly Program	IPC				
	Without Cache	L1i = 16B	L1i = 128B	L1i = 512B	L1i = 1kB
`descending.asm`	0.02493	0.02293	0.08848	0.07804	0.06977
`evenorodd.asm`	0.02439	0.02238	0.02158	0.02083	0.02013
`fibonacci.asm`	0.02482	0.02291	0.05864	0.05313	0.04850
`prime.asm`	0.02463	0.02375	0.05022	0.04576	0.04202
`palindrome.asm`	0.02462	0.02315	0.06947	0.06113	0.05458

Table 2: Throughput (IPC) with `L1d-cache` fixed at 1kB

4.2 Varying Data Cache

Now, we fix the size of the `L1i-cache` at 1kB. We vary the size of the L1d-cache from 16B to 1kB. Plots are included later.

Assembly Program	IPC				
	Without Cache	L1d = 16B	L1d = 128B	L1d = 512B	L1d = 1kB
`descending.asm`	0.02493	0.06490	0.07075	0.07025	0.06977
`evenorodd.asm`	0.02439	0.02013	0.02013	0.02013	0.02013
`fibonacci.asm`	0.02482	0.04903	0.04885	0.04867	0.04850
`prime.asm`	0.02463	0.04202	0.04202	0.04202	0.04202
`palindrome.asm`	0.02462	0.05458	0.05458	0.05458	0.05458

Table 3: Throughput (IPC) with `L1i-cache` fixed at 1kB

5 Graphical Representation

5.1 Varying Instruction Cache

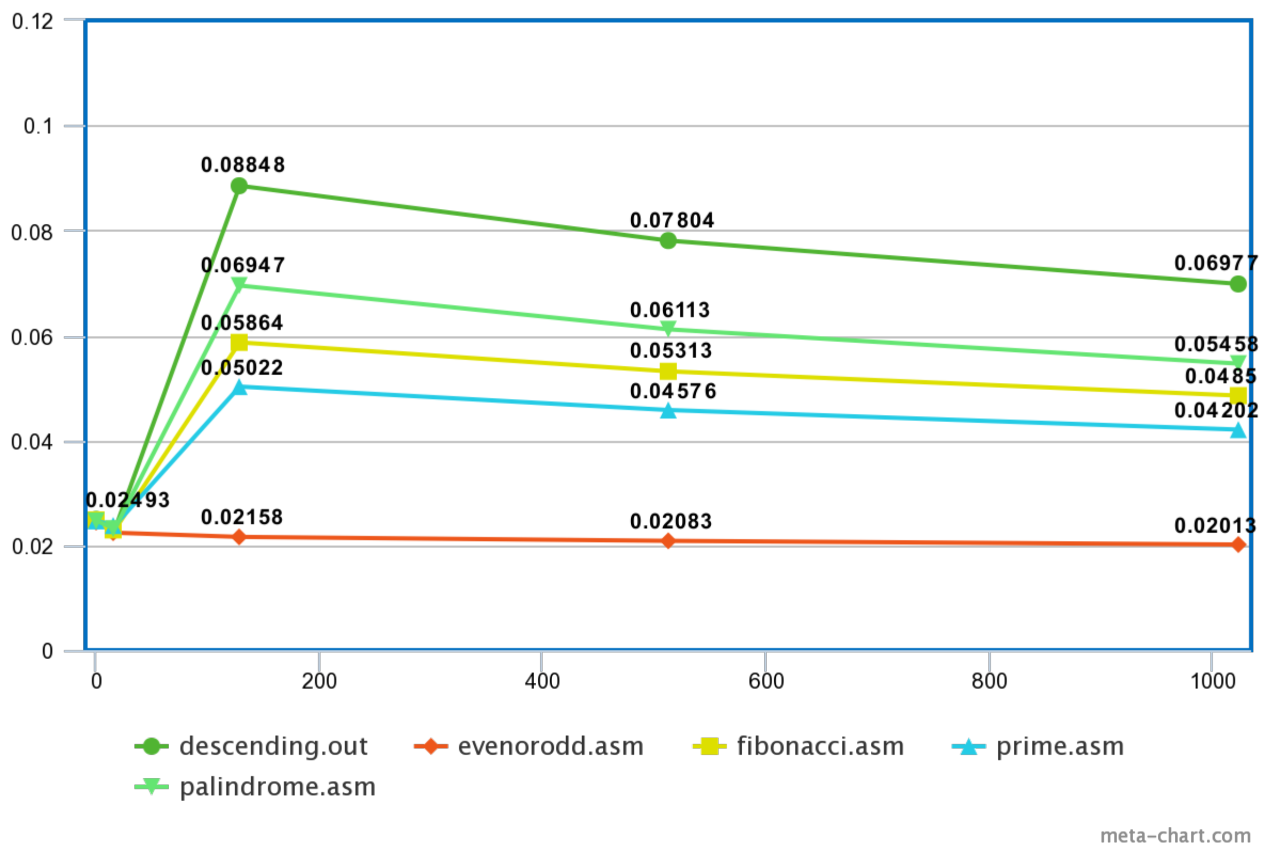

Figure 1: Varying L1i-cache

5.2 Varying Data Cache

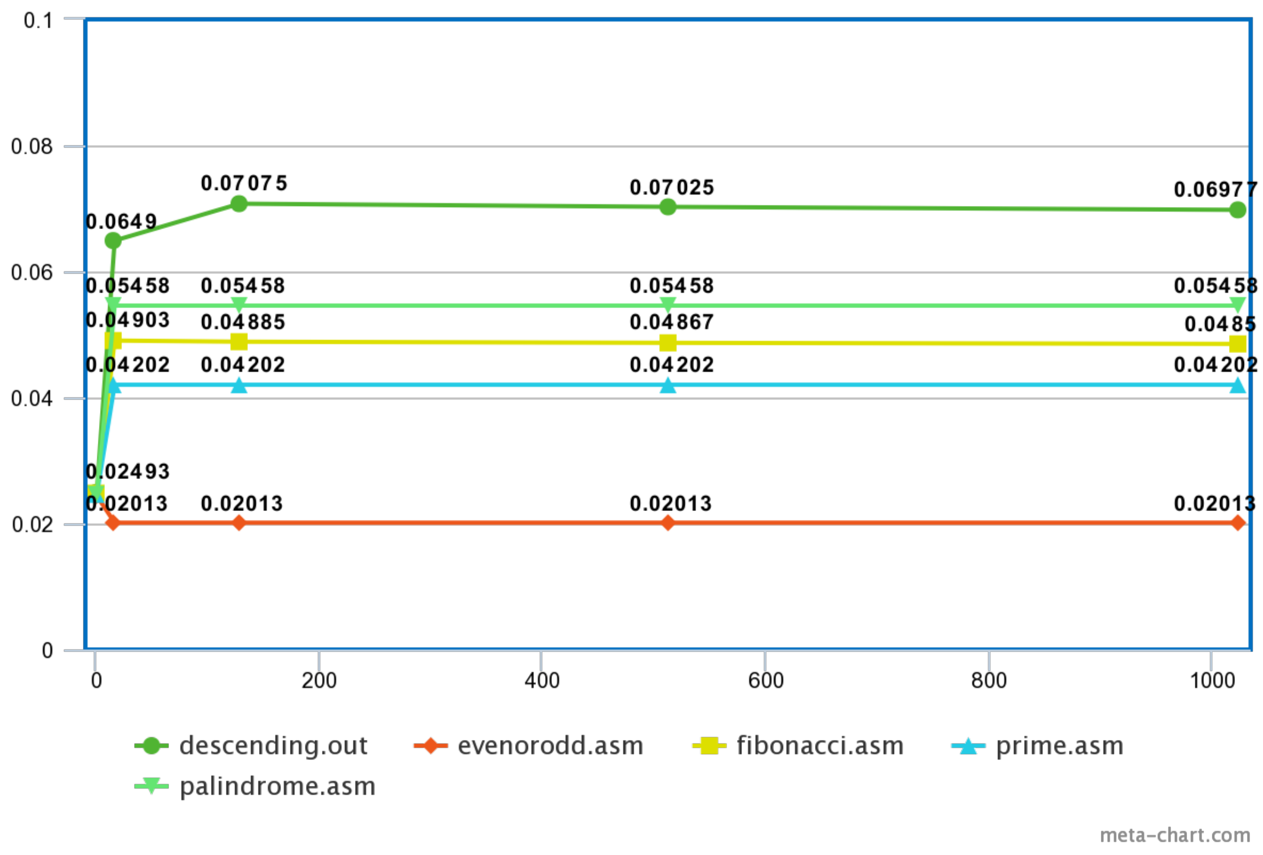

Figure 2: Varying L1d-cache

6 Observations

- Cache latency has been modeled. It is reflected in IF stage and during load/store processes of the MA stage. Separate classes `Cache.java` and `CacheLine.java` were created within `Processor.memorysystem` .
- From the graphs, we observe that introduction of an L1i_cache decreases IPC initially, followed by a sharp increase (except in the case of `evenorodd.asm`). This later steadily decreases with increase in L1i_cache size.
- Also, it could be observed that introduction of an L1d_cache sharply increases IPC initially, followed by tapering off of the IPC on further increase in size of L1d_cache.
- We can conclude that with a considerable amount of data, we can find an optimal amount of L1i_cache and L1d_cache to maximize IPC.