# Assignment 6 Report

Cache Implementation in Processor

April 13, 2025

## Summary

This report discusses the integration of level-1 caches into a processor's memory system. Specifically, two caches are introduced:

- **L1 Instruction Cache (L1i)**: Located between the Instruction Fetch (IF) stage and main memory.
- **L1 Data Cache (L1d)**: Located between the Memory Access (MA) stage and main memory.

These caches are designed using a direct-mapped architecture, with size-dependent latency and basic handling for cache hits and misses. A Java-based simulation model was extended with an event-driven mechanism to emulate memory behavior through cache lines.

## Cache Implementation Highlights

The `Cache` class features:

- Dynamic initialization based on cache size.
- Latency setup using logarithmic logic.
- Handling of read/write cache operations.
- Cache miss delegation to main memory using events.

Example logic used for indexing and tag matching:

```
int indexBits = (int)(Math.log(noOfLines) / Math.log(2));
String addressString = toBinary(address, 32);
int cacheAddress = Integer.parseInt(addressString.substring((32 - indexBits), 32), 2);
```

Currently, cache hits are checked via:

```
if (actualCache[cacheAddress].getTag() == address)
```

This can be improved by comparing only tag bits, excluding the index bits from the address.

# IPC Values for Different Cache Parameters

Table 1: IPC Values for Various Cache Sizes

| Program | No Cache | L1i=16B | L1i=128B | L1i=512B | L1i=1024B | L1d=16B | L1d=128B | L1d=512B | L1d=1024B |
|---|---|---|---|---|---|---|---|---|---|
| descending | 0.009 | 0.015 | 0.017 | 0.018 | 0.017 | 0.016 | 0.018 | 0.020 | 0.017 |
| evenorodd | 0.012 | 0.013 | 0.014 | 0.015 | 0.014 | 0.014 | 0.015 | 0.016 | 0.014 |
| fibonacci | 0.011 | 0.012 | 0.015 | 0.017 | 0.014 | 0.013 | 0.015 | 0.018 | 0.014 |
| palindrome | 0.016 | 0.016 | 0.017 | 0.019 | 0.012 | 0.015 | 0.018 | 0.019 | 0.012 |
| prime | 0.012 | 0.014 | 0.015 | 0.016 | 0.013 | 0.014 | 0.016 | 0.017 | 0.013 |

# Graphs and Observations

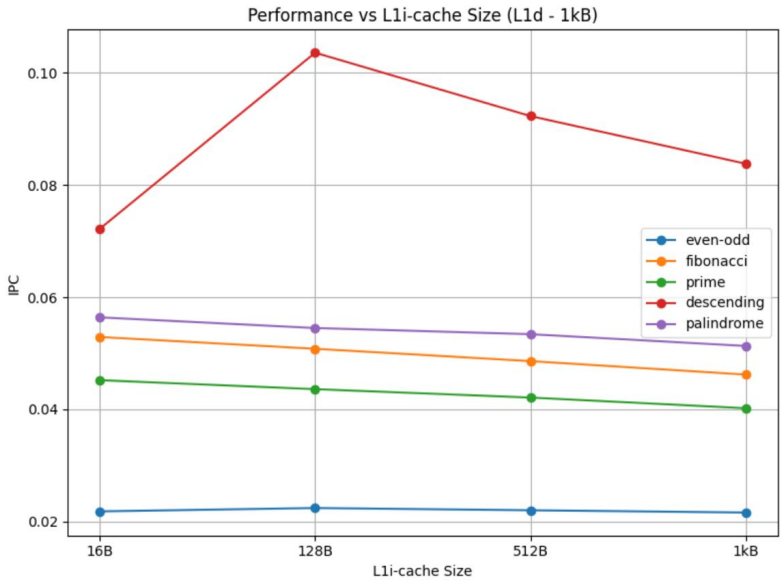


Figure 1: IPC with L1i = 1024B

# Conclusion

Integrating caches in the processor improves performance by reducing memory latency. As observed from the IPC values, increasing the cache size improves IPC until a point, beyond which there is no significant gain. This helps identify an optimal cache configuration. The current direct-mapped model could be enhanced further by experimenting with set-associative and fully-associative caches.

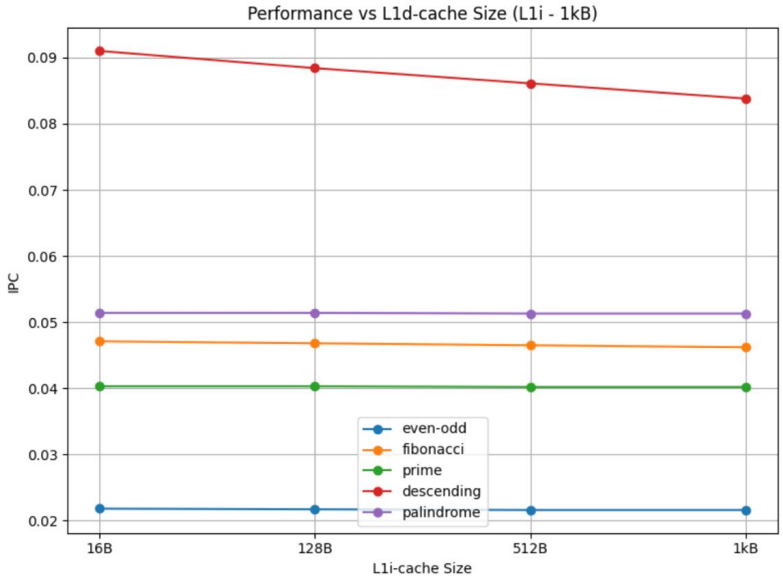


Figure 2: IPC with L1d = 1024B