

CS311: Computer Architecture Lab

Assignment6

Om Deshmukh - 210010033

Tejas Mhaikar - 210030039

October 28, 2023

1 Question 1

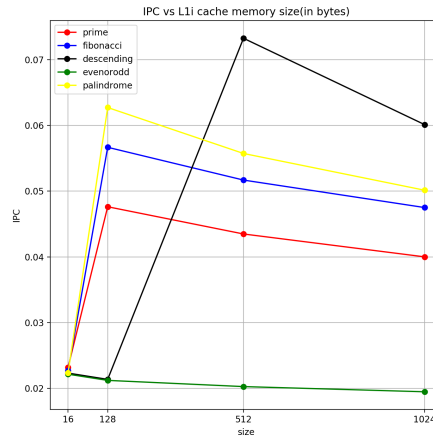


Figure 1: Stats for question 1

Table 1: When L1i cache size is fixed at 1024 Bytes

Benchmark	16 Bytes	128 Bytes	512 Bytes	1024 Bytes
fibonacci.out	0.02319236	0.04761905	0.04347826	0.04
prime.out	0.02268887	0.05666064	0.05167675	0.04749874
descending.out	0.02232143	0.02136752	0.07324005	0.06011199
evenorodd.out	0.0221519	0.02121212	0.02027027	0.01948052
palindrome.out	0.02237315	0.06270997	0.05572139	0.05013429

2 Question 2

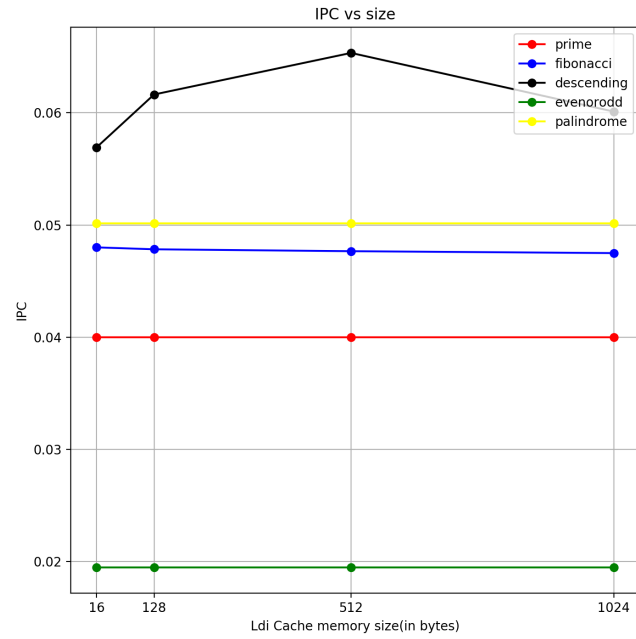


Figure 2: Stats for question 2

Table 2: When L1d cache size is fixed at 1024 Bytes

Benchmark	16 Bytes	128 Bytes	512 Bytes	1024 Bytes
fibonacci.out	0.04	0.04	0.04	0.04
prime.out	0.04749874	0.04783715	0.04766734	0.04800817
descending.out	0.06011199	0.06163458	0.06534035	0.05692452
evenorodd.out	0.01948052	0.01948052	0.01948052	0.01948052
palindrome.out	0.05013429	0.05013429	0.05013429	0.05013429

3 Question 3

Nature of benchmark

For the given benchmarks, at first, our performance (IPC) increases as the cache size increases. This initial boost in performance can be attributed to the larger cache's ability to hold more data which in turn increases the hit-rate, reducing the need to fetch data from slower main memory. As a result, more instructions can be executed without waiting for data to be fetched from the main memory, leading to higher IPC values.

However, later on, as the latency of the cache also increases with the cache size, a point is reached where the trade-off between cache size and latency becomes less favorable. Beyond this point, increasing the cache size doesn't significantly improve performance, and in some cases, it can even lead to a decrease in performance. This happens because the increased cache latency starts to offset the benefits of having a larger cache.

In question 1, where the primary focus is on the L1i-cache, designed for storing instructions, we observe that with larger cache sizes, there is an initial increase in performance for almost every benchmark. This is because a larger L1i-cache can store more program instructions, reducing the number of fetches from slower memory, which, in turn, enhances instruction execution. However, this improvement is not indefinite, and as cache latency increases with larger sizes, performance begins to decrease due to the diminishing returns of cache size expansion.

In question 2, with the primary emphasis on the L1d-cache, designed for data storage, we observe a unique pattern. Performance notably improves, particularly for the "descending" benchmark, as a larger L1d-cache enables more efficient data reuse and reduces the need for data fetches from main memory, aligning with the specific memory access patterns of this benchmark. However, it's important to note that the performance gains level off beyond a 4-byte L1d-cache, indicating that even this relatively modest cache size is adequate for the workloads in the above benchmarks as 'prime.out', 'fibonacci.out', 'palindrome.out' and 'evenorodd.out'.

In summary, the relationship between cache size, cache latency, and performance is complex and depends on the specific characteristics of the benchmarks, as well as the access patterns of both instructions and data in the cache. Balancing cache size and latency is essential to optimize overall system performance for a range of workloads.

4 Question 4

```
.data
a:
100
.text
main:
load %x0, $a, %x4
addi %x4, 10, %x4
addi %x6, 6, %x6
subi %x4, 1, %x4
addi %x7, 2, %x7
addi %x9, 2, %x9
loopi:
subi %x6, 1, %x6
muli %x7, 2, %x7
muli %x7, 2, %x7
muli %x9, 2, %x9
muli %x9, 2, %x9
muli %x7, 2, %x7
muli %x9, 2, %x9
bgt %x6, %x0, loopi
end
```

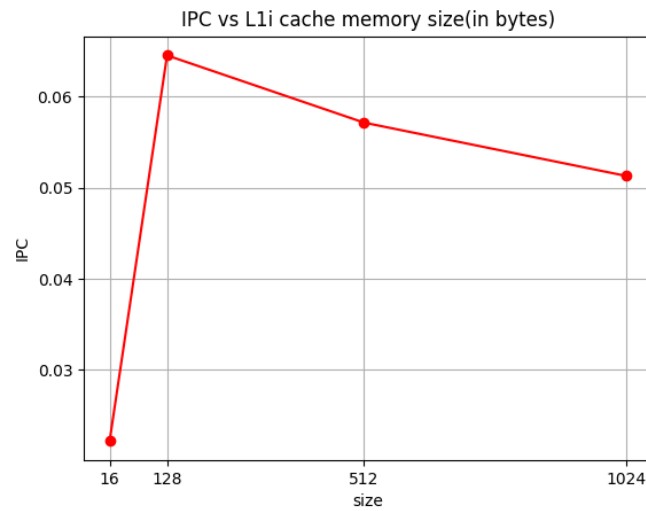


Figure 3: Stats for question 4

5 Question 5

```
                .data
a:
    7
    6
    5
    4
    3
    2
    1
    .text
main:
    addi %x5, 100, %x5
    addi %x10, 200, %x10
    addi %x11, 300, %x11
    addi %x12, 400, %x12
    addi %x13, 500, %x13
    addi %x6, 500, %x6
    addi %x9, 120, %x9
fori:
    store %x9, $a, %x5
    load %x5, $a, %x28
    store %x9, $a, %x10
    load %x5, $a, %x29
    store %x9, $a, %x11
    load %x5, $a, %x30
    store %x9, $a, %x12
    load %x5, $a, %x22
    store %x9, $a, %x13
    load %x5, $a, %x25
    addi %x5, 1, %x5
    blt %x5, %x6, fori
end
```

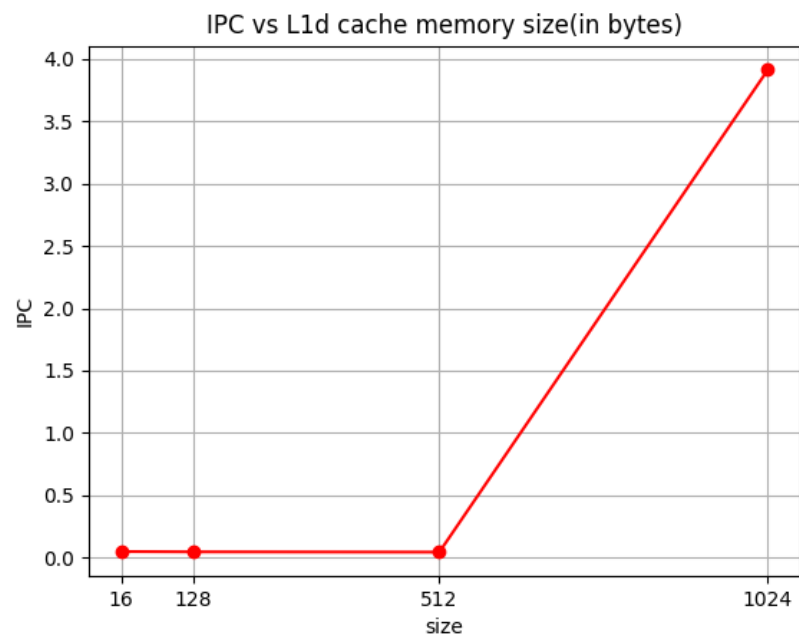


Figure 4: Stats for question 5