Caches Report

8th April, 2024

1. Fixed L1d cache at 1KB

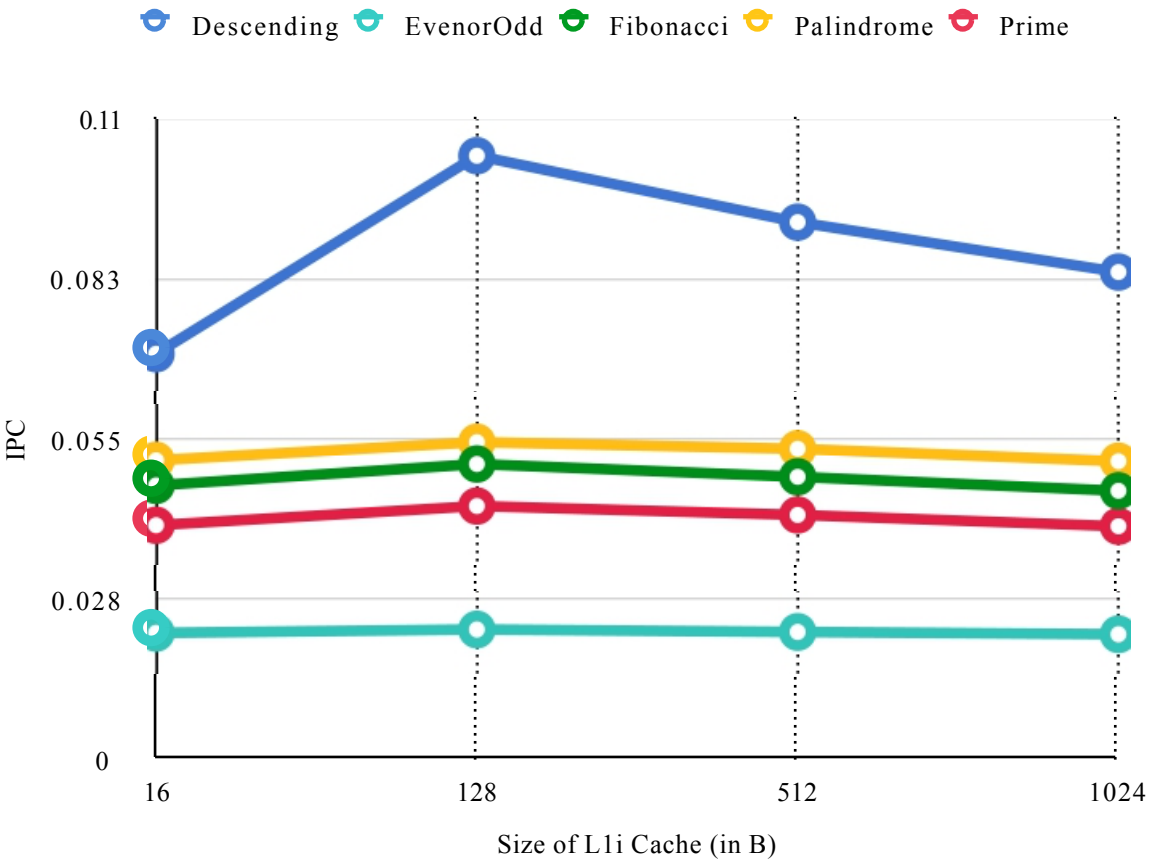

Figure 1: When L1d cache size is fixed at 1024 bytes and L1i-cache is varied

Benchmark	16B	128B	512B	1024B
Descending	0.06984367	0.10362888	0.09230257	0.083787054
EvenorOdd	0.021818181	0.02238806	0.021978023	0.021582734
Fibonacci	0.04712991	0.05078125	0.04859813	0.04620853
Palindrome	0.05141658	0.054505005	0.053376906	0.05125523
Prime	0.040333796	0.043609023	0.042089984	0.040166207

Table 1: IPC vs L1i cache size

2. Fixed L1i at 1KB

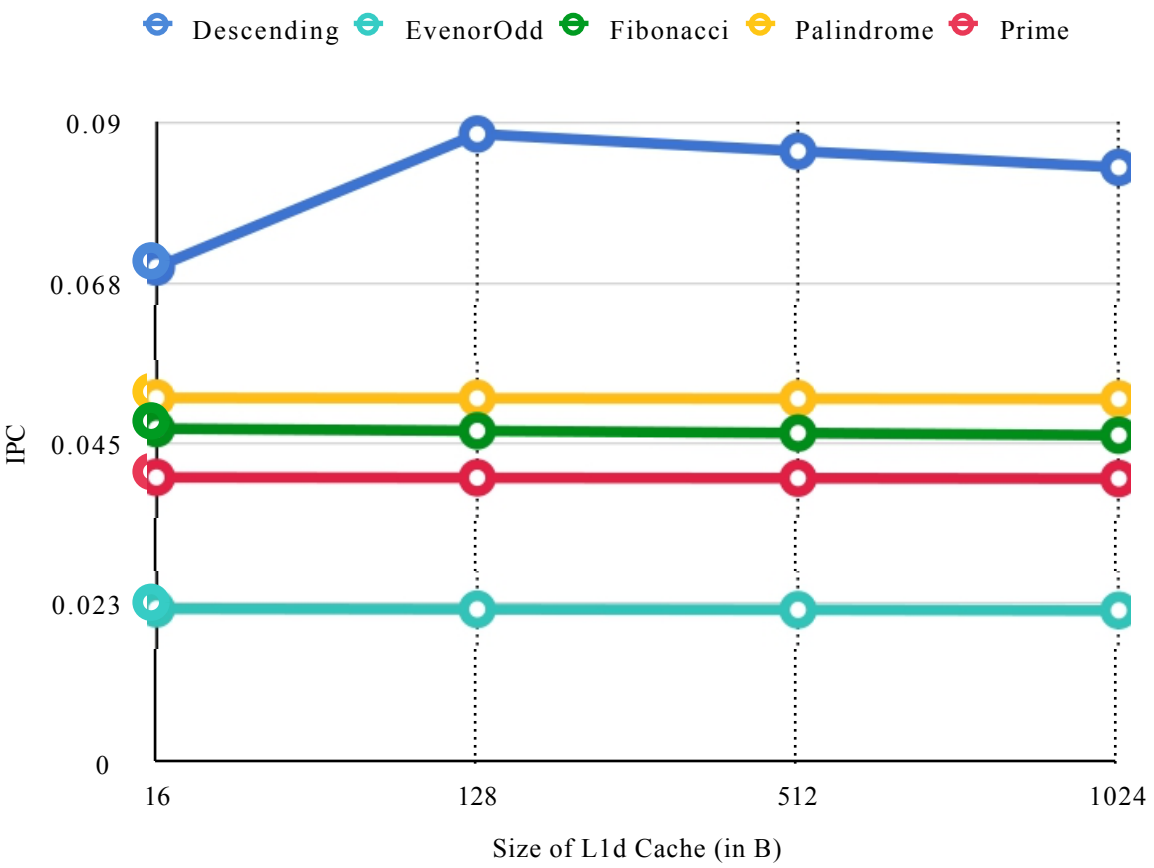

Figure 2: When L1i cache size is fixed at 1024 bytes and L1d-cache is varied

Benchmark	16B	128B	512B	1024B
Descending	0.06984367	0.08844189	0.08605157	0.083787054
EvenorOdd	0.021818181	0.02173913	0.02166065	0.021582734
Fibonacci	0.04712991	0.046818726	0.046511628	0.04620853
Palindrome	0.05141658	0.051362682	0.0513089	0.05125523
Prime	0.040333796	0.04027778	0.040221915	0.040166207

Table 2: IPC vs L1d cache size

3. Observations

Initially, our performance, measured in instructions per cycle (IPC), improves as the cache size increases. This improvement is because larger caches can store more data, leading to a higher hit rate and a reduction in the need to fetch data from slower main memory. Consequently, more instructions can be executed without waiting for data retrieval, resulting in higher IPC values.

However, as cache size increases, the latency of the cache also increases. Eventually, there's a point where the trade-off between cache size and latency becomes less favourable. Beyond this point, further increases in
cache size don't significantly enhance performance and can sometimes even lead to a decrease in performance. This occurs because the increased cache latency starts to counterbalance the advantages of having a larger cache.

In the case of the L1i-cache, which is primarily designed for storing instructions, we see an initial performance boost with larger cache sizes for most benchmarks. This is because a larger L1i-cache can store more program instructions, reducing the need for fetches from slower memory and improving instruction execution. However, this improvement is not indefinite, and as cache latency increases with larger sizes, performance begins to decline due to diminishing returns.

In question 2, focusing on the L1d-cache designed for data storage, we observe a unique pattern. Performance notably improves, especially 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. Nevertheless, it's important to note that the performance

In summary, the relationship between cache size, cache latency, and performance is intricate and depends on the specific characteristics of the benchmarks and the access patterns of both instructions and data in the cache. Finding the right balance between cache size and latency is crucial for optimizing overall system performance across a range of workloads.

4. Evaluation for Benchmark Code-1

```
    .data
a:
    12321
    .text
main:
    load %x0, $a,
    %x3 sub %x7, %x7,
    %x7
loop:
    divi %x3, 10, %x4
    muli %x7, 10, %x7
    add %x7, %x31, %x7
    divi %x3, 10, %x3
    bgt %x3, %x0,
    loop load %x0, $a,
    %x5
    beq %x5, %x7,
    palindrome sub %x10,
    %x10, %x10
    subi %x10, 1, %x10
    end
palindrome:
    sub %x10, %x10, %x10
    addi %x10, 1, %x10
    end
```

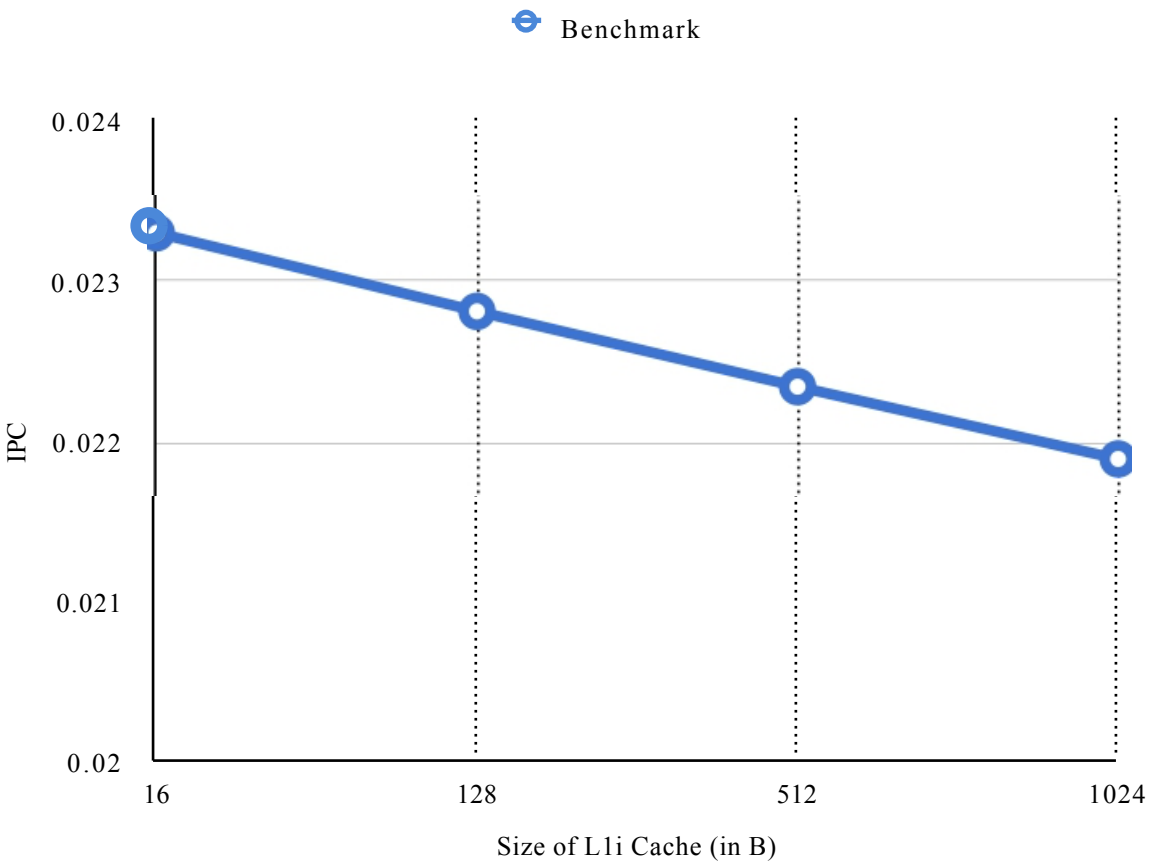

Figure 3: IPC vs L1i Cache Size

5. Evaluation for Benchmark Code-2

Benchmark Code:

```
    .data
a:
    12321
    .text
main:
    load %x0, $a,
    %x3 sub %x7, %x7,
    %x7
loop:
    divi %x3, 10, %x4
    muli %x7, 10, %x7
    add %x7, %x31, %x7
    divi %x3, 10, %x3
    bgt %x3, %x0,
    loop load %x0, $a,
    %x5
    beq %x5, %x7, palindrome
    sub %x10, %x10, %x10
    subi %x10, 1, %x10
    end
```

```
palindrome:
   sub %x10, %x10, %x10
   addi %x10, 1, %x10
   end
```

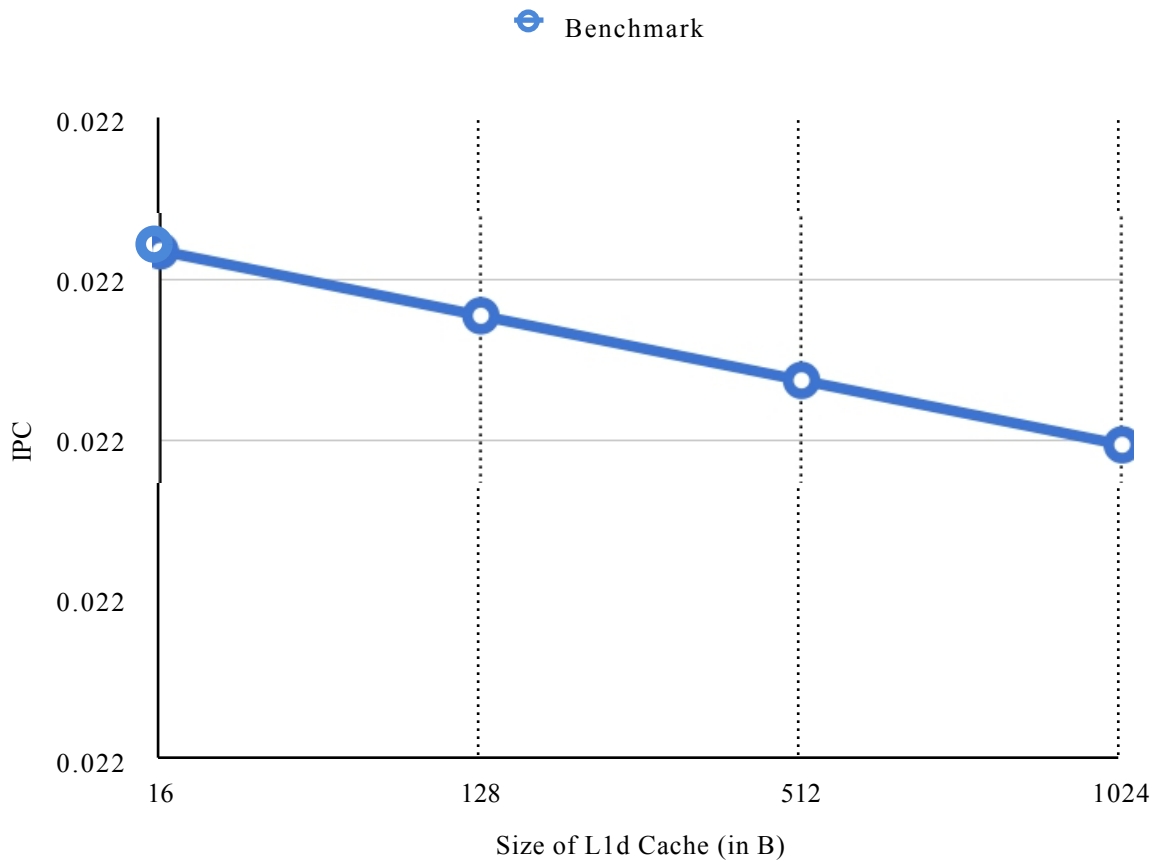

Figure 4: IPC vs L1d Cache Size