Assignment-6 Report
Roll Number: CS23BT071

Overview

The IPC change for evenodd.out is very small. This is because the program only contains 6 dynamic instructions, and each one is accessed just once. Therefore, the cache size has little to no effect on performance.

In contrast, descending.out, fibonacci.out, prime.out, and palindrome.out show a significant increase in IPC when the L1i cache size increases from 16B to 128B. These programs have loops, and the 128B cache is large enough to hold all the loop instructions. However, the 16B cache is too small to fit the entire loop, leading to frequent cache misses and reduced performance.

1 Benchmark Results

L1i	L1d	n	evenodd.out	prime.out	palindrome.out	fibonacci.out	descending.out
16B	1024B	1	0.02280	0.02010	0.02069	0.01933	0.01790
128B	1024B	2	0.02238	0.04315	0.05408	0.05090	0.07428
512B	1024B	3	0.02197	0.04148	0.05196	0.04875	0.09358
1024B	1024B	4	0.02158	0.03978	0.05000	0.04665	0.08575
1024B	16B	5	0.02181	0.03994	0.05015	0.04747	0.04570
1024B	128B	6	0.02173	0.03994	0.05015	0.04707	0.09153
1024B	512B	7	0.02166	0.03983	0.05005	0.04684	0.08947
1024B	1024B	8	0.02158	0.03978	0.05000	0.04665	0.08575

Table 1: Updated IPC Values for Object Files with Varying L1i and L1d Cache Sizes

2 Graphs

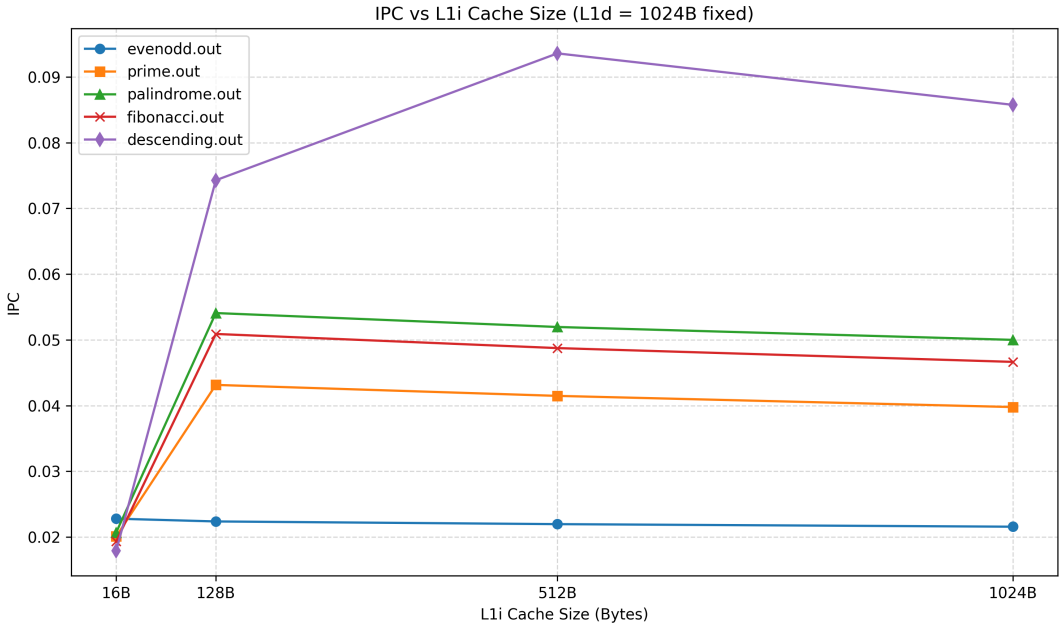

Figure 1: First x the size of the L1d-cache at 1kB. Vary the size of the L1i-cache from 16B to 1kB (remember to change latency accordingly), and study the performance (instructions per cycle)

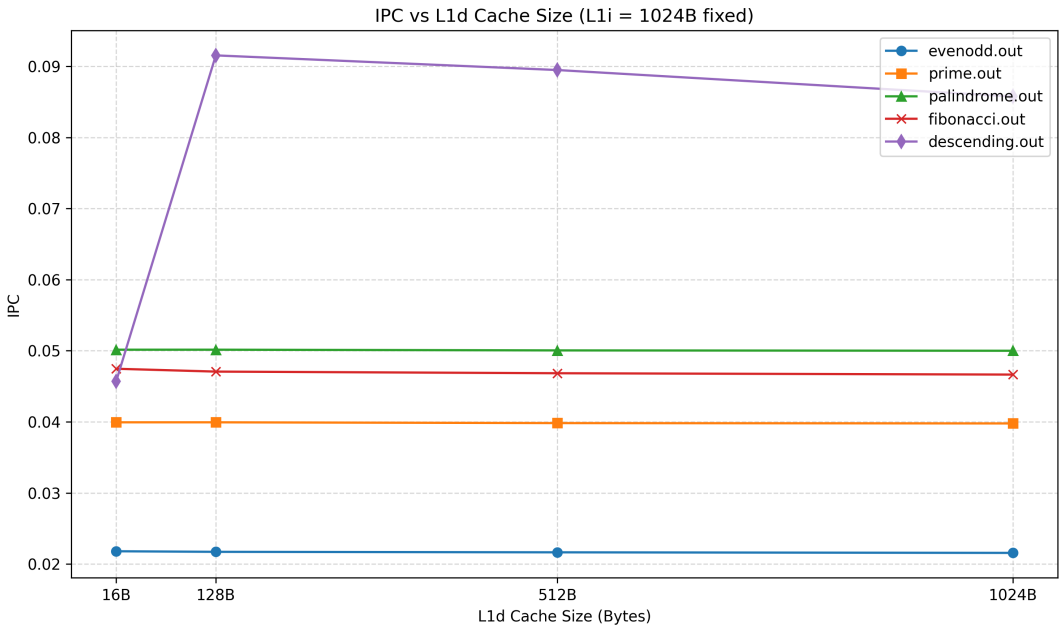

Figure 2: Now x the size of the L1i-cache at 1 kB. Vary the size of the L1d-cache from 16B to 1kB.

3 Performance Improvement

For Toy benchmarks we have which find's greatest improvement. The results for L1d = 1024kB:
Q)4
1) L1i = 16kB, IPC = 0.033
2) L1i = 128kB, IPC = 0.0736
Q)5
For L1i = 1024kB:
1) L1d = '16kB, IPC = 0.0913
2) L1d = 128kB, IPC = 0.1584