CS311: Computer Architecture Lab

Assignment 6 - Report

Anudeep Tubati 170010039
S V Praveen 170010025

The aim was to implement Caches for Instructions and Data (and their respective latencies) while developing a pipelined processor Simulator for ToyRISC.

Table: *L1i = 1kB, L1d from 8B to 1kB (IPC is listed in each cell)*

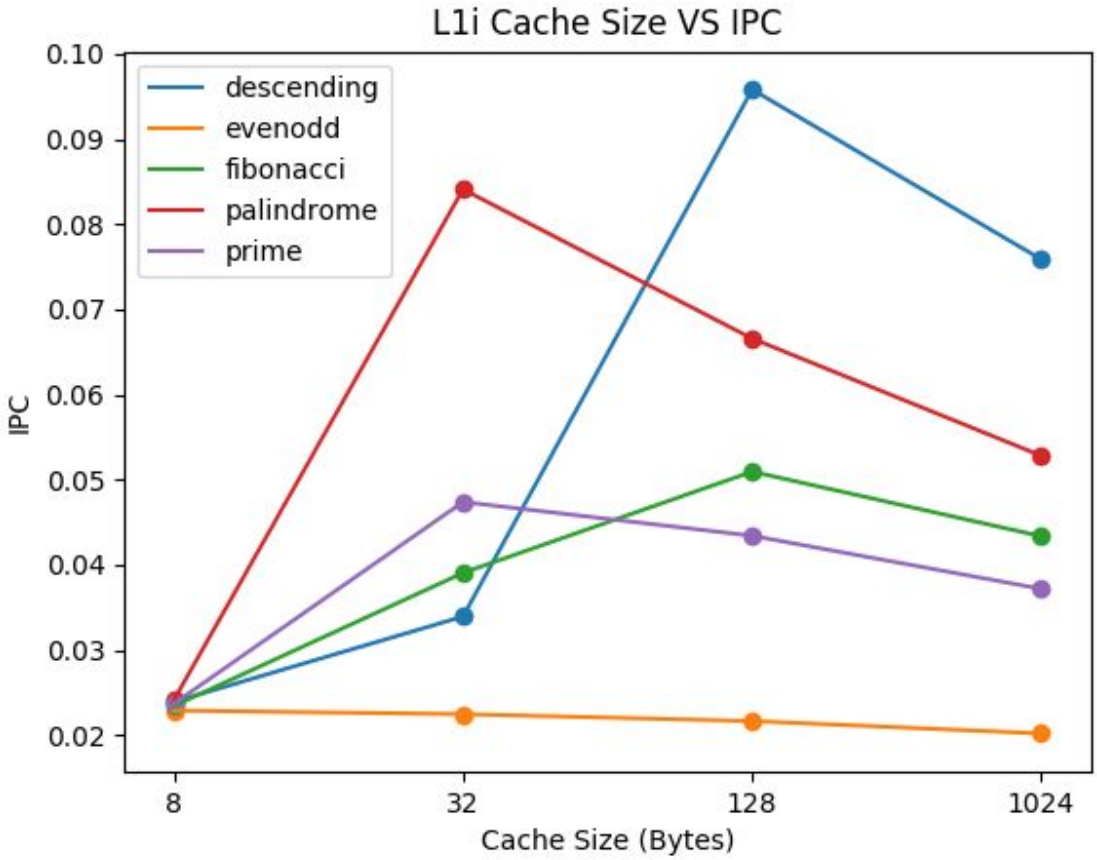
Sl No.	***Program***	***8B***	***32B***	***128B***	***1024B***
1.	*descending.asm*	*0.058512*	*0.082147*	*0.081855*	*0.075869*
2.	*even-odd.asm*	*0.020408*	*0.020408*	*0.020408*	*0.020202*
3.	*fibonacci.asm*	*0.044879*	*0.044699*	*0.044343*	*0.043357*
4.	*palindrome.asm*	*0.052972*	*0.052972*	*0.052972*	*0.052801*
5.	*prime.asm*	*0.037323*	*0.037323*	*0.037323*	*0.037179*

Table: *L1d = 1kB, L1i from 8B to 1kB*

Sl No.	***Program***	***8B***	***32B***	***128B***	***1024B***
1.	*descending.asm*	*0.023918*	*0.033934*	*0.095814*	*0.075869*
2.	*even-odd.asm*	*0.022900*	*0.022471*	*0.021660*	*0.020202*
3.	*fibonacci.asm*	*0.023423*	*0.039058*	*0.050947*	*0.043357*
4.	*palindrome.asm*	*0.024114*	*0.084097*	*0.066576*	*0.052801*
5.	*prime.asm*	*0.023692*	*0.047385*	*0.043413*	*0.037179*

Clearly, as cache size increases, IPC tends to increase followed by a decrease. The correlation depends on the nature of the program and while it is negative for some, they also resemble the nature of a bell curve for others.

Varying Instruction Size Cache vs IPC

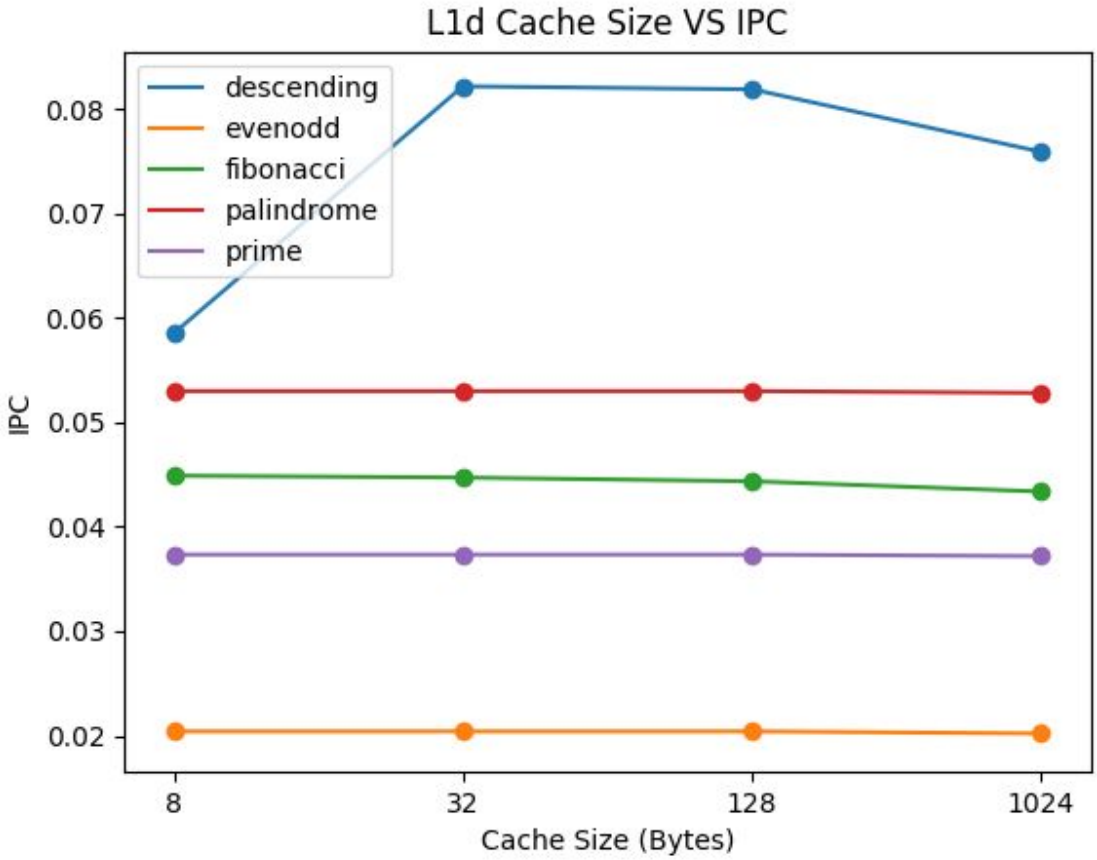


➔ Here, nearly all benches exhibit a similar nature. It is important to note the peaks for each program as those points are where the program runs most effectively through the pipelined processor with the highest IPC.

➔ There is an initial tendency for IPC to increase as expected with increasing cache size. However, further increasing the cache size decreases IPC due to higher cache latencies thus decreasing IPC.

➔ Clearly, programs with iterative loops exhibit higher temporal locality among instructions and increased cache sizes help them execute better.

Varying Data Size Cache vs IPC



- ➔ The IPC is nearly constant for those programs which do not repeatedly access data from memory.
- ➔ Descending seems to be the only program with repeated calls to memory.
- ➔ Increasing cache size from 8B to 32B shows a significant performance increase owing to higher cache hit ratio.
- ➔ However, increasing cache size more doesn't really improve IPC due to higher cache latencies that render the cache hits less effective.

Toy Benchmark Program for L1i from 32B to 128B is fibonacci_4.asm(included in zip)
[The point was to get more than 8 instructions called consecutively using a for loop]

Results

For 32B, IPC = 0.022908

For 128B, IPC = 0.031868

Clearly, the 128B Cache outperforms the 32B due to an increased number of lines, although it had higher latency.

Note: The 32B Cache can show better performance than the 128B cache when less than 8 instructions are executed iteratively using a for loop.

Toy Benchmark Program for L1d from 32B to 128B is testbench2.asm(included in zip)
[The point was to get more than 8 memory access instructions called consecutively using a for loop]

For 32B, IPC = 0.023323

For 128B, IPC = 0.068254

Clearly, the 128B Cache outperforms the 32B due to an increased number of lines, although it had higher latency.

Note: The 32B Cache can show better performance than the 128B cache when less than 8 load/store instructions are executed iteratively using a for loop.