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.

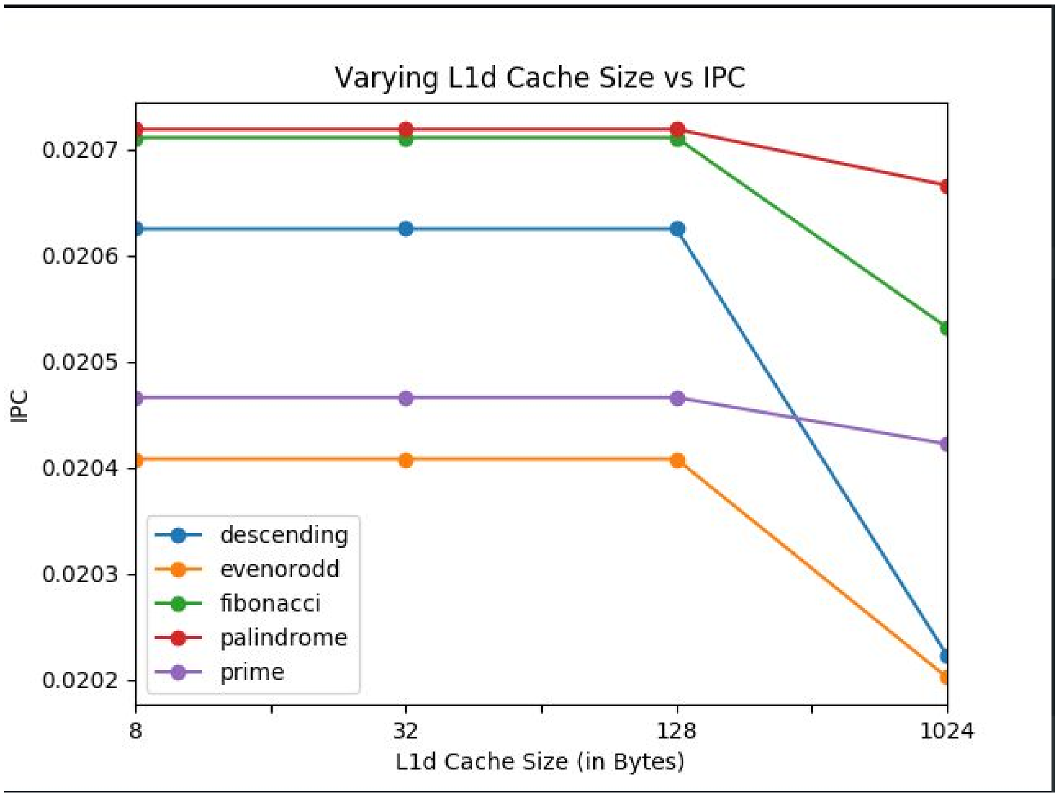
Table: *L1i = 1kB, L1d from 8B to 1kB (IPC is listed in each cell)*

Sl No.	***Program***	***8B***	***32B***	***128B***	***1024B***
1.	*descending.asm*	*0.020625*	*0.020625*	*0.020625*	*0.020223*
2.	*even-odd.asm*	*0.020408*	*0.020408*	*0.020408*	*0.020202*
3.	*fibonacci.asm*	*0.020711*	*0.020711*	*0.020711*	*0.020532*
4.	*palindrome.asm*	*0.020719*	*0.020719*	*0.020719*	*0.020666*
5.	*prime.asm*	*0.020466*	*0.020466*	*0.020466*	*0.020422*

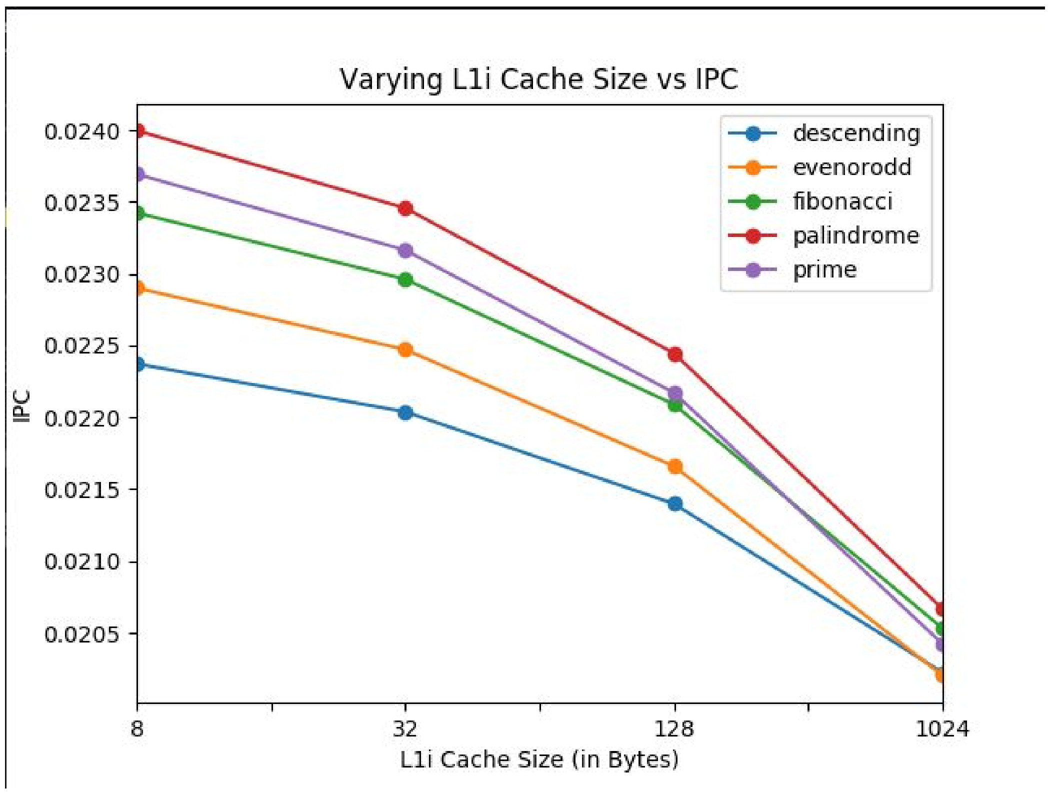
IPC Statistics - Table: *L1d = 1kB, L1i from 8B to 1kB*

Sl No.	***Program***	***8B***	***32B***	***128B***	***1024B***
1.	*descending.asm*	*0.022372*	*0.022038*	*0.021398*	*0.020223*
2.	*even-odd.asm*	*0.022900*	*0.022471*	*0.021660*	*0.020202*
3.	*fibonacci.asm*	*0.023423*	*0.022961*	*0.022090*	*0.020531*
4.	*palindrome.asm*	*0.023996*	*0.023456*	*0.022446*	*0.020666*
5.	*prime.asm*	*0.023692*	*0.023163*	*0.022171*	*0.020422*

Clearly as cache size increases, IPC tends to decrease. Hence they are negatively correlated.



- We see the slope of prime is lesser than descending. This owes to the fact that descending makes the most number of memory access calls as compared to other programs.
- The slope of the curve is directly proportional to the number of load/store instructions called.
- The IPC is nearly constant for smaller cache sizes as the increased cache sizes are rendered ineffective by their higher latencies.



- Here, nearly all benches exhibit similar slope. This proves how effective an increase in instruction cache size can be for most of the programs.
- Clearly, programs with iterative loops exhibit higher temporal locality among instructions and increased cache sizes help them execute better. An example of this is ‘descending’ whose performance stays nearly the best always.

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 = 0.0229087

For 128B, IPC = 0.022066

Clearly, the 128B Cache outperforms the 32B due to 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 testbench.asm(included in zip)
[The point was to get more than 8 memory access instructions called consecutively using a for loop]

For 32B, IPC = 0.021117

For 128B, IPC = 0.020799

Clearly, the 128B Cache outperforms the 32B due to 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.