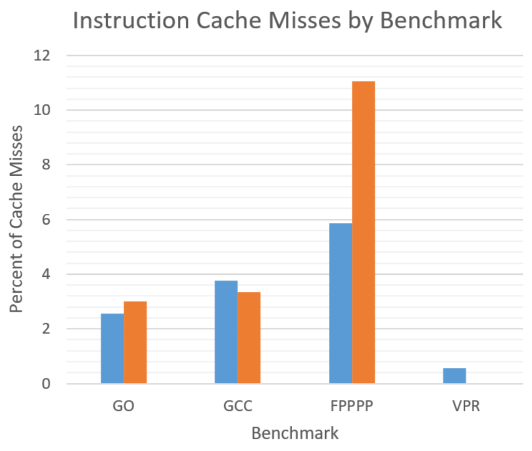
1 Results

Table 1: Cache Miss Percentage by Type and Benchmark

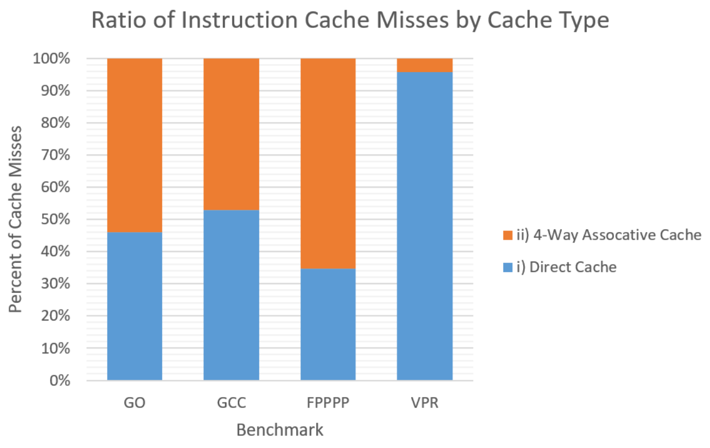
	GO	GCC	FPP	VPR
icache_direct_miss	2.5605	3.7723	5.8665	0.5611
icache_setassoc_miss	3.0019	3.351	11.0606	0.0254
dcache_load_miss	1.8655	2.0537	0.4416	2.7919
dcache_store_miss	0.9732	0.8236	0.1659	0.8622
dcache_wback	6.4684	4.687	1.2806	7.7286
icache_pref_miss	0.3937	1.0505	0.0479	0.0029

2 Analysis

2.1 Instruction Cache



(a) Absolute Values

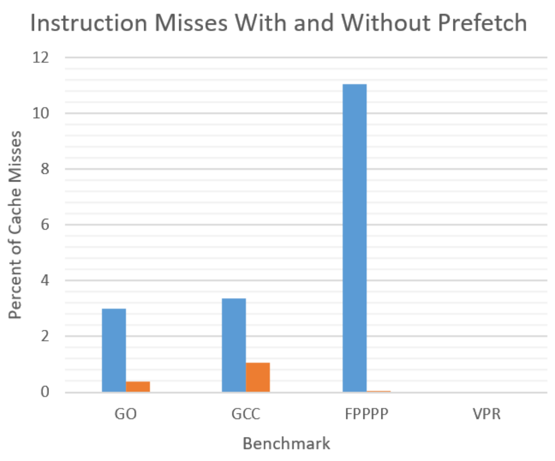


(b) Relative Values

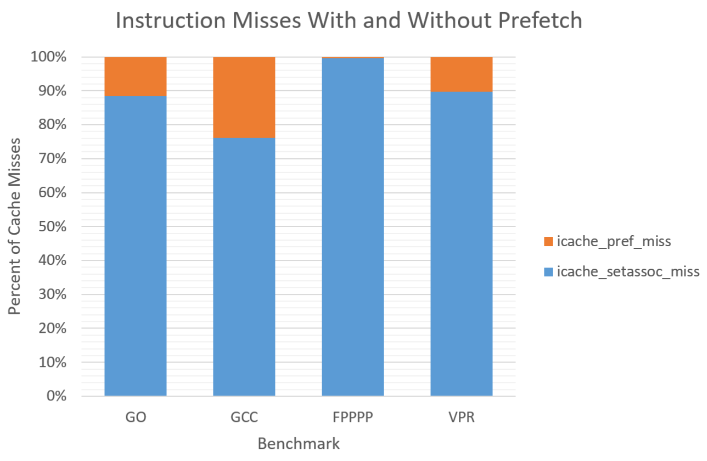
Figure 1: Percent of Instruction Cache Misses by Benchmark and Cache Type

GCC and VPR predictably have better performance with 4-way associative cache than the direct-mapped cache because conflict misses are mitigated. However FPPPP and GO have better performance with direct cache. This indicates that the direct-mapped cache does not have as many conflict misses for these programs. Perhaps this is because FPPPP and GO are more iterative programs than GCC and VPR, thus there are less unique instructions that need to be executed.

FPPPP has the most instruction cache misses without prefetches, in part because FPPPP is one of the longer benchmarks, which means there are the most cold-start (compulsary) cache misses. However, with prefetch it has by far the most performance improvement. This is beacause floating-point programs have have longer looping structures and and less



(a) Absolute Values



(b) Relative Values

Figure 2: Percent of Instruction Cache Misses With and Without Prefetch

if-else statements [1], thus most prefetched instructions are accurate because there are fewer branches.

2.2 Data Cache

FPPP has the least overall amount of data cache misses and VPR has the most. This is because FPPPP has very good temporal locality because floating-point programs have fewer branches and spend most of the time in loops, whereas VPR and the next-worst performing benchmark GCC, both have poor temporal locality [2].

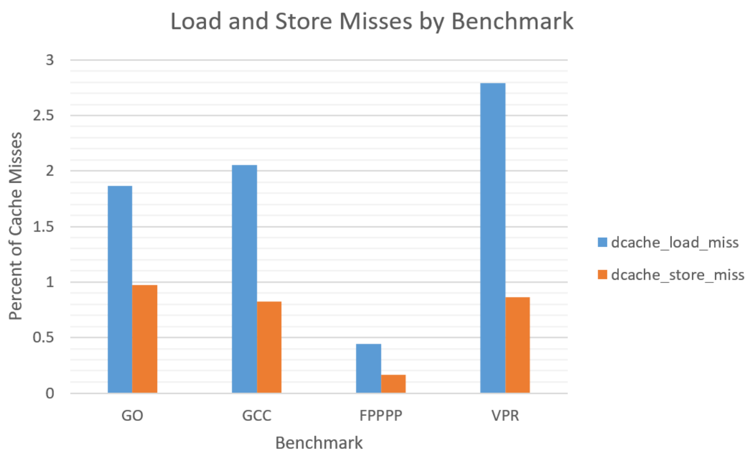


Figure 3: Percent of Load/Store Cache Misses

The percent of load misses is always higher than store misses. This is because there is simply a higher percentage of load instructions than store instructions in each benchmark.

References

[1] C.-C. Cheng, "The schemes and performances of dynamic branch predictors," Berkeley Wireless Research Center, Tech. Rep., 2000.

[2] A. Joshi, A. Phansalkar, L. Eeckhout, and L. K. John, "Measuring benchmark similarity using inherent program characteristics," *IEEE Trans. Computers*, vol. 55, no. 6, pp. 769–782, 2006.