COE1541 Project 2 Analysis

Tim Cavrak, Jonathan Gramley, Dan Mingey

We tested our simulator by setting the associativity of all the caches to 1 (direct mapped). We simulated cache misses, hits, and eviction by writing and reading from addresses that map to the same indices. For example, setting the address in a lw instruction to 0 and then another lw instruction with an address of 1024 for a 1KB size cache would both map to the same index. We wrote one test called test1.tr that covered most of the cases. This test case covers instruction cache misses and data cache hit (cycle 402), and an eviction in L2 that causes a write to the write buffer (adds additional 60 cycles at the end of the program while write buffer is still writing back to memory).

```
 ** opening file test1.tr
[cycle 335] LOAD: (PC: 0)(sReg_a: 0)(dReg: 0)(addr: 20)//L1 Data miss
[cycle 402] STORE: (PC: 4)(sReg_a: 0)(sReg_b: 0)(addr: 20)//L1 Data Hit
[cycle 469] STORE: (PC: 8)(sReg_a: 0)(sReg_b: 0)(addr: 24)// L1 Data Miss
[cycle 542] LOAD: (PC: 12)(sReg_a: 0)(dReg: 0)(addr: 1044)// L1 Data Miss/ Uses Write
Buffer
[cycle 615] LOAD: (PC: 16)(sReg_a: 0)(dReg: 0)(addr: 1048)// L1 Data Miss/ Uses Write
Buffer
+ Simulation terminates at cycle : 675
I-cache: 5 accesses, 5 misses, miss rate =  1.00 //Block size 1 so each instruction causes
                                                 //instruction miss in L1 and L2
D-cache: 5 Reads, 4 Read misses, Read miss rate =  0.80
L2-cache: 9 Reads, 9 Read misses, Read miss rate =  1.00 //no associativity in L2

** opening file test2.tr
[cycle 137] LOAD: (PC: 0)(sReg_a: 0)(dReg: 0)(addr: 32)
[cycle 204] LOAD: (PC: 4)(sReg_a: 0)(dReg: 0)(addr: 32)
[cycle 205] LOAD: (PC: 8)(sReg_a: 0)(dReg: 0)(addr: 32)
[cycle 206] LOAD: (PC: 12)(sReg_a: 0)(dReg: 0)(addr: 32)
[cycle 207] LOAD: (PC: 16)(sReg_a: 0)(dReg: 0)(addr: 32)
+ Simulation terminates at cycle : 206
I-cache: 5 accesses, 2 misses, miss rate =  0.40 //block size 16 allows spacial locality to lower
                                                 //miss rate
D-cache: 5 Reads, 1 Read misses, Read miss rate =  0.20 //multiple reads of the same value
L2-cache: 3 Reads, 3 Read misses, Read miss rate =  1.00
```

** opening file test3.tr
[cycle 335] LOAD: (PC: 0)(sReg_a: 0)(dReg: 0)(addr: 20)
[cycle 402] STORE: (PC: 4)(sReg_a: 0)(sReg_b: 0)(addr: 20)
[cycle 469] LOAD: (PC: 8)(sReg_a: 0)(dReg: 0)(addr: 1044)
[cycle 470] STORE: (PC: 12)(sReg_a: 0)(sReg_b: 0)(addr: 1044)
[cycle 471] LOAD: (PC: 16)(sReg_a: 0)(dReg: 0)(addr: 20)
+ Simulation terminates at cycle : 470
I-cache: 5 accesses, 5 misses, miss rate = 1.00
D-cache: 5 Reads, 2 Read misses, Read miss rate = 0.40
L2-cache: 7 Reads, 7 Read misses, Read miss rate = 1.00

Experiment 1:

sample_large1.tr

Miss rate (after warm up)																		
	Direct map									4-way associative								
	1KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)			1 KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)		
	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2
4B	0.19	0.45	0.38	0.01	0.37	0.71	0.00	0.33	0.58	0.00	0.36	0.76	0.00	0.30	0.80	0.00	0.26	0.71
16B	0.08	0.43	0.48	0.01	0.31	0.59	0.00	0.26	0.46	0.02	0.31	0.56	0.00	0.22	0.72	0.00	0.18	0.59

Execution cycles (after warm up)																		
	Direct map									4-way associative								
	1KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)			1 KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)		
	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2
4B	1151367526			829232828			587789013			977700970			830559305			645688564		
16B	935137830			585755629			393421519			723585201			573150809			408143039		

Remarks & Explanation

Based on the results in sample_large1.tr, there are several noticeable conclusions to be made. When only changing the block size from 4B to 16B, there was almost always a reduction in the miss rate for the Instruction Cache and Data Cache. The L2 Cache had a higher miss rate, but this was because the increase in block size leads to less frequent calls to memory. Therefore, the overall cycle number decreased when increasing the block size from 4B to 16B.

For the most part, the 4-way associative caches had a longer cycle number. This is due to the L2 cache miss rate increasing when switching to 4-way associativity, which was caused by an increase in write backs. This means that the reuse of instructions was less common.

For the most part, increasing the size of the caches decreased the miss rate and the execution time in general. This is because the caches were able to hold more data, therefore had to evict less data which leads to less calls to memory.

sample_large2.tr

Miss rate (after warm up)																		
	Direct map									4-way associative								
	1KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)			1 KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)		
	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2
4B	0.07	0.62	0.73	0.00	0.59	0.86	0.00	0.55	0.86	0.01	0.61	0.94	0.00	0.58	0.85	0.00	0.53	0.87
16B	0.02	0.61	0.84	0.00	0.57	0.92	0.00	0.54	0.90	0.00	0.58	0.94	0.00	0.55	0.94	0.00	0.54	0.88

Execution cycles (after warm up)																		
	Direct map									4-way associative								
	1KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)			1 KB L1 (8KB L2)			4KB L1 (32KB L2)			16KB L1 (128KB L2)		
	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2	I	D	L2
4B	1843175126			1576283667			1500034014			2359232517			1980739025			1736804735		

16B	1780231079	1665501996	1608350047	2241274584	2099833970	1860445213

Remarks & Explanation

Based on the results from sample-large2.tr, when increasing the block size from 4B to 16B, there were varying results in the miss rates of the caches but the L2 cache almost always increased leading to an increase in the execution time. This is most likely due to the fact that large block size led to more write backs which evicted larger amounts of data raising the miss rate and the calls to memory.

For the most part, the 4-way associative caches had a longer cycle number. There was not a general trend for the miss rates when switching from the 4 way associative cache to the direct mapped cache so the cycle increase is due to increased memory accesses in the 4-way case. This means that the reuse of instructions was less common.

For the most part, increasing the size of the caches decreased the miss rate and the execution time in general. This is because the caches were able to hold more data, therefore had to evict less data which leads to less calls to memory.

Experiment 2:

Block size 16B and L2 size 128KB
Cost = S * (1 + 0.1 * A)

We have two arguments: if we weight performance equal to cost, then a 1KB and direct mapped instruction and data cache is most efficient because there is not a huge increase factor in performance when increasing size and associativity. The cost increases when increasing the size and associativity, but the performance benefits do not match the cost increases. However, if we weighted performance with a factor of K, where K is the ratio of how much more the performance matters than cost, the most efficient result changes. We chose a value of K=100, meaning the performance is 100x more valuable than the cost. This led to a result of the most efficient setup having a 4-way associative and 2KB block size cache for both the instruction cache and data cache.

Instruction Cache Iterative Analysis

Instruction Cache		sample_large1.tr					
Size	Associativity	Cost	Performance (cycles)	performance benefit non-weighted	Average benefit non-weighted	performance benefit weighted	average benefit weighted
1	1	1.1	482112754	0	0	-99	-99
2	1	2.2	465806272	0.9649929961	0.9776429486	-101.5007004	-100.2357051
4	1	4.4	438090619	2.899513632	2.942674703	-106.0486368	-101.7325297
8	1	8.8	437940808	6.899137177	6.942486608	-102.0862823	-97.75133925
16	1	17.6	433173767	14.88702227	14.93640422	-95.29777256	-90.35957782
1	4	1.4	444485053	0.1880726803	0.2279813289	-107.192732	-103.2018671
2	4	2.8	433565387	1.433482122	1.484824354	-108.6517878	-103.5175646
4	4	5.6	433465067	3.978679316	4.027701631	-106.1320684	-101.2298369
8	4	11.2	433198308	9.068903507	9.118270095	-101.1096493	-96.17299045
16	4	22.4	433080618	19.25041925	19.29998726	-90.95807464	-86.00127384
			min=	0	0	-108.6517878	-103.5175646
Instruction Cache		sample_large2.tr					
Size	Associativity	Cost	Performance (cycles)	performance benefit non-weighted		performance benefit weighted	
1	1	1.1	1650009767	0		-99	
2	1	2.2	1634146941	0.9902929011		-98.97070989	
4	1	4.4	1626965066	2.985835774		-97.41642261	
8	1	8.8	1626965489	6.985836038		-93.41639624	
16	1	17.6	1626885492	14.98578617		-85.42138307	
1	4	1.4	1642066606	0.2678899775		-99.21100225	
2	4	2.8	1634825574	1.536166586		-98.38334136	
4	4	5.6	1626931508	4.076723946		-96.32760539	
8	4	11.2	1626937358	9.167636684		-91.23633163	
16	4	22.4	1627098441	19.34955527		-81.04447303	
			min=	0		-99.21100225	

Data Cache Iterative Analysis:

Data Cache		sample_large1.tr					
Size	Associativity	Cost	Performance (cycles)	performance benefit non-weighted	average benefit non-weighted	performance benefit weighted	average benefit weighted
1	1	1.1	482112754	0	0	-99	-99
2	1	2.2	465998079	0.9654190098	0.980721934	-101.458099	-99.9278066
4	1	4.4	454105259	2.938323782	2.965443752	-102.1676218	-99.4556248
8	1	8.8	446800875	6.920967301	6.955789055	-99.90326989	-96.42109446
16	1	17.6	441026778	14.90684018	14.94753759	-93.31598217	-89.24624105
1	4	1.4	451405792	0.2047020889	0.2370002137	-105.5297911	-102.2999786
2	4	2.8	437602247	1.443740015	1.491712719	-107.6259985	-102.8287281
4	4	5.6	429770381	3.969117606	4.025866962	-107.0882394	-101.4133038
8	4	11.2	422974465	9.042002912	9.106823276	-103.7997088	-97.31767236
16	4	22.4	416009378	19.2047376	19.27748331	-95.52624004	-88.25166867
			min=	0	0	-107.6259985	-102.8287281
Data Cache		sample_large2.tr					
Size	Associativity	Cost	Performance (cycles)	performance benefit non-weighted		performance benefit weighted	
1	1	1.1	1650009767	0		-99	
2	1	2.2	1643476714	0.9960248582		-98.39751418	
4	1	4.4	1637830405	2.992563722		-96.74362779	
8	1	8.8	1634661618	6.99061081		-92.93891903	
16	1	17.6	1630823134	14.988235		-85.17649993	
1	4	1.4	1644371326	0.2692983386		-99.07016614	
2	4	2.8	1640545261	1.539685423		-98.03145767	
4	4	5.6	1636439148	4.082616318		-95.73836825	
8	4	11.2	1633390766	9.171643641		-90.8356359	
16	4	22.4	1628180207	19.35022903		-80.97709729	
			min=	0		-99.07016614	