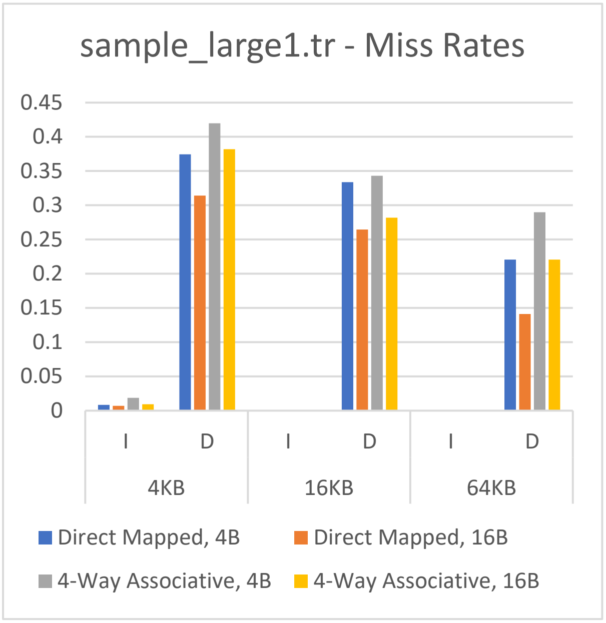
1541 Intro to Computer Architecture: Project 2

Group Members: Austin Linder, Matthew Salemi, Kevin Carr

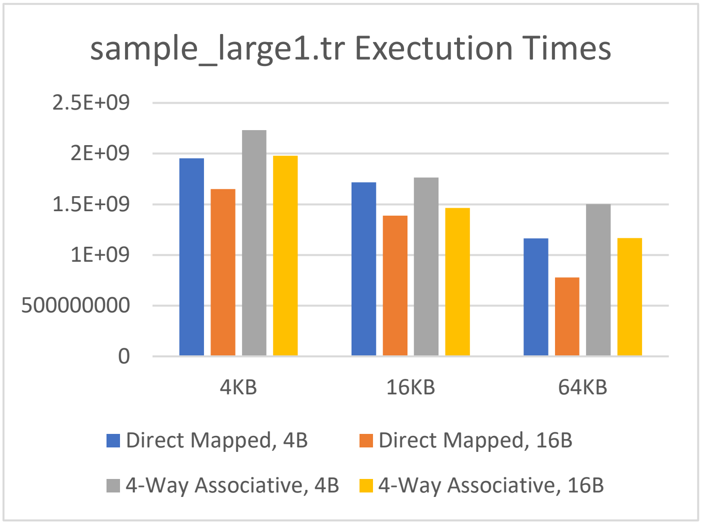
*Our project uses full stalling of the pipeline on any cache miss. That is, the cycle number is incremented to simulate a full stoppage of the pipeline until any miss in the cache returns data or an instruction.

Experiment 1:

Sample_large1.tr - Miss Rates						
	Direct Map					
	4KB		16KB		64KB	
	I	D	I	D	I	D
4B	0.008642	0.3743	5.556e-5	0.3337	1.780e-5	0.2208
16B	0.00691	0.3141	1.812e-5	0.2647	4.804e-6	0.1412
	4-Way Associative					
	4KB		16KB		64KB	
	I	D	I	D	I	D
4B	0.01882	0.4197	1.780e-5	0.3431	1.780e-5	0.2901
16B	0.009374	0.3818	5.274e-6	0.2817	4.804e-6	0.2206



Sample_large1.tr - Total Execution Cycles			
	Direct Mapped		
	4KB	16KB	64KB
4B	1954118075	1716115515	1162665595
16B	1650908395	1382068075	779835355
	4-Way Associative		
	4KB	16KB	64KB
4B	2231263675	1763864955	1502614475
16B	1978817995	1465500395	1167075435



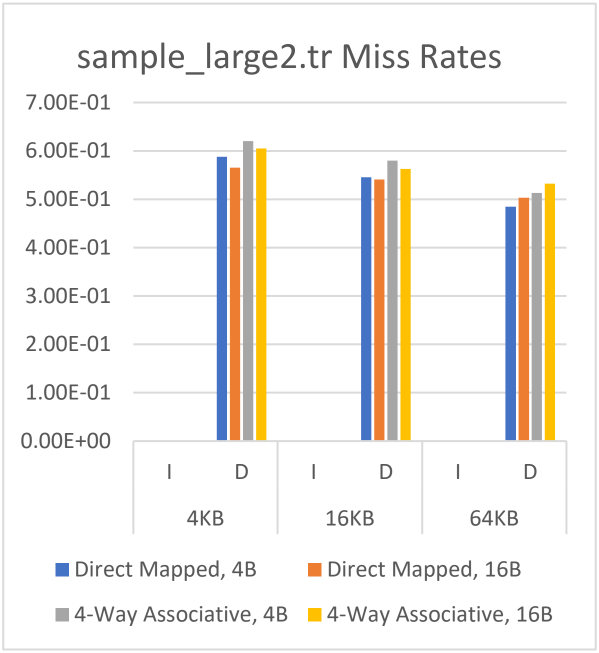
Findings:

In the longer traces, the instruction cache missed much less than the data cache. Likely because instructions are being repeated in loops, as well as functions. Larger cache sizes vastly decreased the miss rate in the instruction cache by several place values. This trend was also noticeable in the data cache but to a much lower degree. The decrease in miss rate can be attributed to less collisions due to a larger number of hashable indexes. Total cycle time also decreased when the size of the cache increased. This is because the miss rate decreased, while the miss penalty stayed the same. This would not be the case in a real cache, where it would take more time to search through a larger cache.

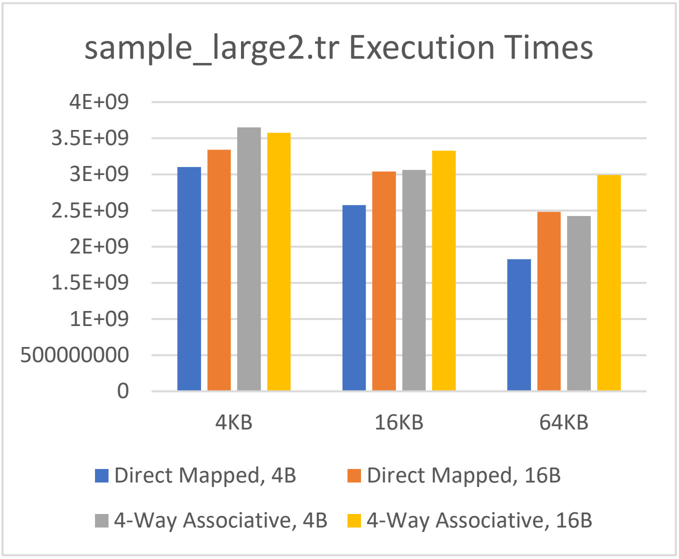
Associativity seems to have a mostly negative effect on both the miss rate and the total number of cycles. We expected that when the associativity was higher, the miss rate would decrease, however our data did not back up our prediction. We weren't able to figure out why our prediction and the results didn't align, but we suspect that there is something in the 'cache.h' file that we overlooked and is causing these issues, if any.

A larger block size decreased the miss rate and total cycle count for many of the experiments. The exception is that total execution cycles increased with block size with the direct mapped caches on sample large 2. Pulling more data into the cache at once definitely saved time by causing us to miss less and avoid reaching into secondary memory less often.

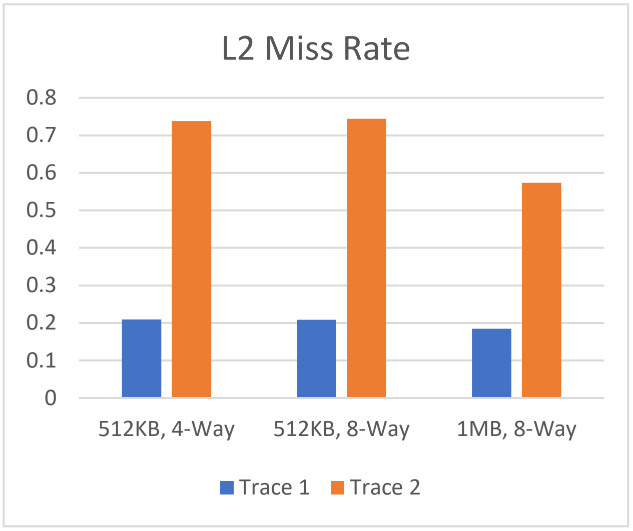
Sample_large2.tr – Miss Rates						
	Direct Map					
	4KB		16KB		64KB	
	I	D	I	D	I	D
4B	5.606e-5	0.5879	5.904e-6	0.5453	5.904e-6	0.4848
16B	1.796e-5	0.5653	1.569e-6	0.5410	1.569e-6	0.5030
	4-Way Associative					
	4KB		16KB		64KB	
	I	D	I	D	I	D
4B	5.904e-6	0.6204	5.904e-6	0.5802	5.904e-6	0.5128
16B	1.569e-6	0.6051	1.569e-6	0.5629	1.569e-6	0.5325

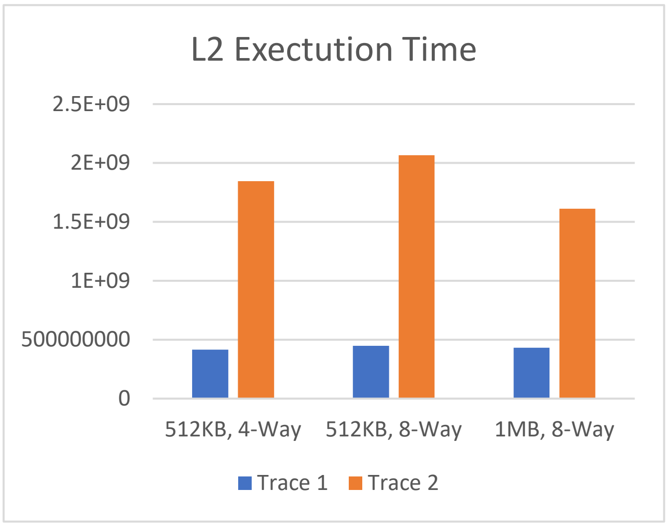


Sample_large2.tr – Execution Time			
	Direct Mapped		
	4KB	16KB	64KB
4B	3100969355	2573746235	1829292715
16B	3342154875	3039354555	2483921995
	4-Way Associative		
	4KB	16KB	64KB
4B	3649087835	3060527755	2423110635
16B	3577136555	3328524075	2992607515



Experiment 2:





Findings:

	L2 Miss Rate		Total Execution Time	
	Trace 1	Trace 2	Trace 1	Trace 2
512KB, 4-Way	0.2093	0.7380	414757805	1844992985
512KB, 8-Way	0.2084	0.7439	447965961	2066107269
1MB, 8-Way	0.1848	0.5731	430900059	1612970779

In experiment 2, there is a significant decrease in the miss rate when a larger cache size is used. It should also be noted that there was an increase in the miss rate when the n-way cache was larger indicating that due to there being less hashable indexes, more instructions were hashed into the same index and more overwrites occurred. There is a balance between how many times an instruction key may be hashed and how many available slots in our cash can fill it. In the case of experiment 2, the most efficient cache was the 1MB cache that used 8-way associativity because it struck the right balance of cache storage to hashed instructions and data.

Concerns:

There was an increase in the miss rate as associativity increased, which wasn't expected. Looking at the 'cache.h' file, we couldn't find anything in the implementation that would make that happen.

In the first experiment on the second trace, we expected lower execution times due to larger block sizes. We observed lower miss rates on bigger block sizes, but the execution times went up, which was not what we were expecting.