William Swiss, Andrew Baxter
Computer Organization Spring 2017
Final Project: MIPS trace profiler

The contributions of the team members include:
Andrew: design and implementation of the cache and relevant pipeline-cache interactions and some stalls in the pipeline push procedure;
William: implementation of the rest of the pipeline stalls, documentation, and compiling testing data for the report.
The source is provided with the document, and all the data used and produced is also available here: https://github.com/willdathrill/rpi_comp_org_project.
Below is a chart with our results:

Index Bits	Block size	Cache Associativity	Branch prediction	Cycle count	CPI	Correct Branch Predictions	Cache misses	Cache hits	Cache miss rate
7	1	1	0	48547	1.396	5765	1390	34473	3.87%
7	1	1	1	53033	1.526	1279	1390	34473	3.87%
6	1	2	0	40556	1.167	5764	502	35361	1.40%
6	1	2	1	45040	1.296	1280	502	35361	1.40%
5	1	4	0	39309	1.131	5760	363	35500	1.01%
5	1	4	1	43785	1.260	1284	363	35500	1.01%
6	2	1	0	47757	1.374	5754	1301	34562	3.62%
6	2	1	1	52221	1.503	1290	1301	34562	3.62%
5	2	2	0	39261	1.130	5754	357	35506	.99%
5	2	2	1	47325	1.258	1290	357	35506	.99%
4	2	4	0	37848	1.089	5754	200	35663	0.56%
4	2	4	1	42312	1.218	1290	200	35663	0.56%
6	4	1	0	42979	1.237	5753	770	35093	2.15%
6	4	1	1	47441	1.365	1291	770	35093	2.15%
5	4	2	0	37345	1.075	5753	144	35719	0.40%
5	4	2	1	41807	1.203	1291	144	35719	0.40%
4	4	4	0	36751	1.057	5753	78	35785	0.22%
4	4	4	1	41213	1.186	1291	78	35785	0.22%

While we were not able to match the given output cycle-for-cycle, we were able to get out cycle count within .38% of the count given in the sample output. This sample output specified a block size of two 32-bit words, a 2-way set associative cache, and predicted all branches taken.
Of the nine cache configurations with a size of under 10 Kibibits, the best cache performance was observed with a 128-bit block size and a 4-way set associative cache (the last row in the table). The run which predicted branches not taken had an additional CPI increase of 12% over the one which predicted branches taken.

This result makes sense, given the nature of the program. Matrix multiplication relies heavily on repeating the same blocks code, and operates on contiguous data. The large block size increases hit rate by preloading the loop's consecutive instructions and the next elements of the matrix. The high associativity reduces the chance that this repeated code and data is evicted. However, we were surprised that the branches-not-taken predictor outperformed the branches-taken predictor due to the same abundance of repeated code, meaning more taken branches.

Overall, this project was helpful in elucidating the inner workings of MIPS processors and the performance impact of the many design choices that can be made in cache specification. Also, the benefits of simulating a simple processor and instruction set for workable data were demonstrated: greater knowledge of CPU architecture increases awareness of the lower-level problems for real-world programs in the future, even outside of the MIPS architecture. The concepts like cache properties (block size, associativity, etc.) and pipelining affect the functioning of modern processors and virtual machines.