Arden Diakhate-Palme (aqd)
Carlos Armendariz (carmenda)
Ankita Bhanjois (abhanjoi)

Lab 4: REPORT_cyc

1. **Report counter statistics for the 3 benchmark inputs: benchmarks/mixed.c, benchmarks/mmmRV32IM.c, and benchmarksO3/mmmRV32IM.c**

	mixed.c		mmmRV32I		O3/mmmRV32I	
Num cycles Elapsed	41103		70360		25231	
Num Instr Fetched	32245		42761		19690	
Num Instr Exec'd	30711		42101		19588	
	FWD | rd==x1	**BKWD | rd != X1**	**FWD | rd==x1**	**BKWD | rd != X1**	**FWD | rd==x1**	**BKWD | rd != X1**
Num br executed:	2203	3700	16	4912	16	560
Num br taken:	159	3660	1	4606	0	525
Num br taken correct pred:	13	3607	0	4592	0	516
Num JAL executed:	0	340	0	3	0	3
Num JAL without BTB pred:	0	0	0	0	0	0
Num JAL correct pred:	0	0	0	0	0	0
Num JALR executed:	299	88	6	0	4	0
Num JALR without BTB pred:	271	271	0	0	4	4
% BR Correct	0.6132		0.9318		0.8958	
Number of Cache Hits:	7236		15759		7515	
Number of Cache Misses:	26		1685		597	

2. **Discuss and report additional counters you introduced**
    a. We added performance counters to track data cache hits and misses in order to come up with ways to possibly optimize our cache and find possible bugs in our implementation.
3. **Describe the specific optimization techniques/mechanisms you employed. Use the counter data to support their effectiveness. (How does it improve performance? What execution conditions or program behaviors are being exploited? How commonplace is the exploited condition or behavior?)**
    a. We added a cache which capitalized on both the spatial and temporal locality of the matrix multiplication programs and improved the execution times. Since mmmRV32IM.c often makes calls to LW from usually sequential memory locations (since we are scanning through all values of two matrices). We are not directly capitalizing on spatial locality because our block size is only 2 words. However, we are making 4 additional memory accesses to sequential addresses for each D-MEM access, so we make use of spatial locality in this manner (since the ensuing addresses don't all map to the same cache block).

4. **Discuss if the observed behavior and performance agree with your intuition/expectations. (Does your technique/mechanism work as well as you had hoped? What is the performance bottleneck at the end?)**
    a. Our cache performed quite well with 11073 hits and 1473 misses on the matrix multiplication program. If we had increased the associativity of our cache, we could perhaps decrease the miss-rate, but implementing an a-way set-associative cache would have been tricky. Our performance bottleneck now is the cache miss rate and the BTB miss rate, both of which incur significant delays. We could address both by fine-tuning the data cache and the BTB to the matrix multiplication benchmark.