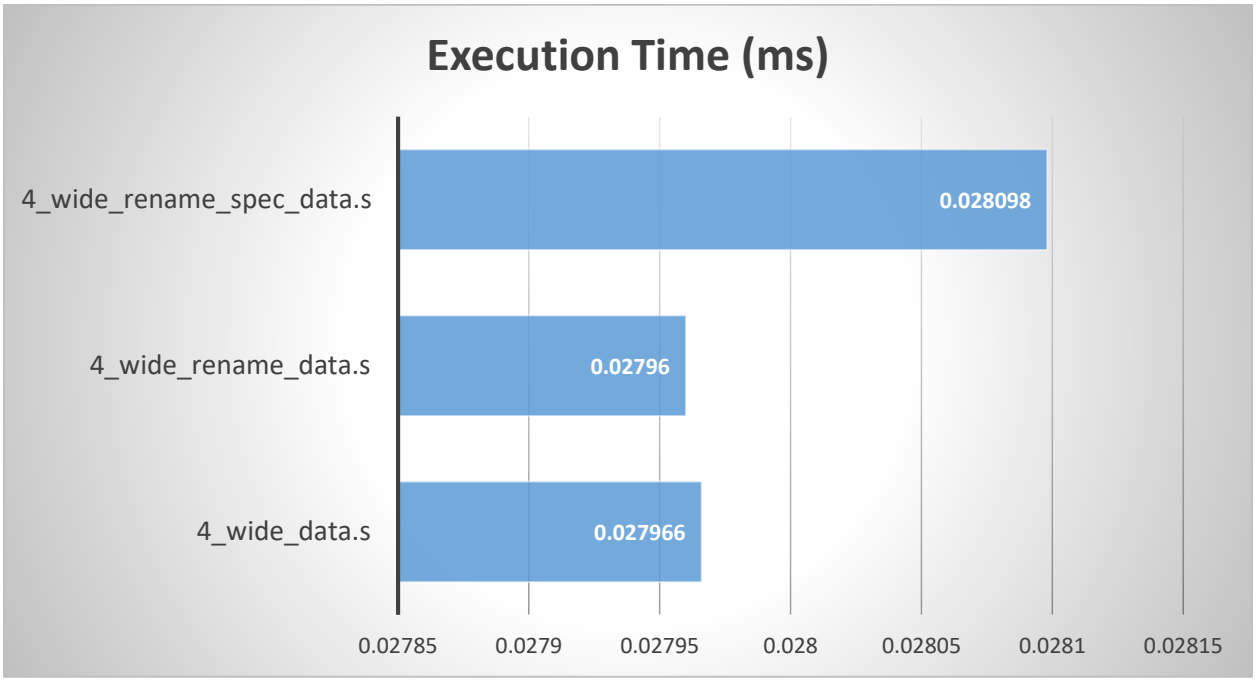
10/22/2018

VLIW Speculation Report

Project 1, Part 2

Andrew Tam, Victor Keyes, Chris Church
TEAM 29

Performance Results



Instruction Set	Cycles	Execution Time (ms)	Speculative Loads
4_wide_data.s	13983	0.027966	N/A
4_wide_rename_data.s	13980	0.02796	N/A
4_wide_rename_spec_data.s	14049	0.028098	4

Discussion

For this implementation, hardware pipelining was not used, and it was assumed that stores and loads used different functional units. As the raw data shows above, the renamed instructions out performed normal 4-wide instructions without renaming. This is expected because renaming only offers improvements without penalties. The renamed instructions also performed better than the renamed, speculatively loaded instructions. The reason for this has to do with the potential improvement per speculative load compared to the potential penalty per speculative load. Since speculative loads are used to hide the load latencies, and they are often hidden in the latencies of the store whose access they are speculating, the store latency of 1 in this implementation doesn't provide much cover. As for the penalty, in the worst case scenario where the load and the store happen at the same time, and there is a collision, the penalty is the latency of the load. In this implementation, the penalty will be 20 cycles. Since this set of instructions only has 4 speculative loads, if even one collision occurs, the execution time will be worse than not speculatively loading.