	Cycles with original code	Cycles with Licm+DCE+CP	Precent Speedup
examples/test/licm/licm.bril	77	68	11.688
benchmarks/ackermann.bril	1464231	1464231	0.000
benchmarks/collatz.bril	169	169	0.000
benchmarks/gcd.bril	46	46	0.000
benchmarks/sqrt.bril	322	293	9.006
benchmarks/binary-fmt.bril	100	100	0.000
benchmarks/digital-root.bril	248	248	0.000
benchmarks/fib.bril	121	120	0.826
benchmarks/loopfact.bril	116	80	31.034
benchmarks/perfect.bril	26	26	0.000
benchmarks/recfact.bril	104	72	30.769
benchmarks/sum-bits.bril	73	73	0.000
benchmarks/check-primes.bril	8468	5034	40.553
benchmarks/eight-queens.bril	1006454	961666	4.450
benchmarks/fizz-buzz.bril	259	168	35.135
benchmarks/mat-mul.bril	1990407	1992907	-0.126
benchmarks/sum-sq-diff.bril	3039	1726	43.205

Overall, the results is positive with the following optimizations: loop invariant code motion (LICM), dead code elimination (DCE) and Constant Propagation (CP). As the complier currently does not support the ability to merge blocks, therefore in some cases, LICM will degrade of performance due to the introduction of blocks which results in additional copy instructions. Some of the benchmark suit contains unsupported extensions and is excluded from the reported results. Furthermore, some results are excluded as the LICM code fails to terminate when there is a nest loop (known bug).