	Time (seconds)	instructions	Rel to Start	Rel to Prev	Improvement Notes
midmark	**3.88**	21384996334	1	1	**No improvement - Base Case**
sandmark	**63.74**		1	1	
midmark	2.38	20749618819	0.613	0.613	Compiled with optimization turned on and linked against -lcii-O1
sandmark	62.48		0.98	0.98	
midmark	2.34	20752481013	0.603	0.983	Compiled with optimization turned on and linked against -lcii-O2
sandmark	59.69		0.936	0.955	
midmark	1.05	5377409740	0.271	0.449	Took out the bitpack getu (large percent) and changed it to be inline bitshifts
sandmark	27.9		0.437	0.467	
midmark	0.88	488169586	0.227	0.838	Replaced segment_load and segment_store with inline operations (also took out a couple of unnecessary dereferences in our initial instruction reading for loop)
sandmark	23.63		0.37	0.847	
midmark	0.81	5474818344	0.209	0.92	Replaced our switch statement with conditional statements.
sandmark	21.8		0.342	0.922	
midmark	0.77	5425818717	0.198	0.951	Realized our final project had huge memory leaks (half) and just by fixing those were able to drop another few seconds.
sandmark	18.84		0.296	0.864	
midmark	0.65	4657547576	0.167	0.844	After experimenting with different starting sizes for our large segment arrays. We found that the larger the value, the faster the program speed. Therefore, we decided to make the largest array possible and started with that. This dropped our time significantly.
sandmark	15.59		0.245	0.827	
midmark	0.64	4639133309	0.165	0.984	Removed our 14 opcode functions and just placed them within our big conditional statement. Didn't change much.
sandmark	15.64		0.245	1.003	
midmark	0.66	4633079327	0.17	1.03	Got rid of Bitpack totally. inlined bitpack newu. again, turned out worse actually.
sandmark	15.95		0.25	1.0198	
midmark	0.63	4614546325	0.162	0.954	Optimized for loops, removing unnecessary dereferences to things by storing static local variables outside of loops.
sandmark	15.51		0.243	0.972	
midmark	0.63	4550338140	0.162	1	Fixed memory non-leaking memory errors in valgrind. We weren't sure if there would be any effect. Very insignificant effect.
sandmark	15.48		0.243	0.998	
midmark	**0.57**	4221139831	0.147	0.905	Got rid of Hanson's sequence one and for all. Changed our freed_mem sequence to a static array. Good riddance.
sandmark	**14.06**		0.22	0.908	