Benchmark	Time	Instructions	Relative to Start	Relative to Prev.	Improvement Made	Comment
sandmark	352.068	-	1.0000	1.0000		
advent	123.178	-	1.0000	1.0000		
midmark	14.145	1.08E+11	1.0000	1.0000	No improvement (starting point)	
sandmark	286.01	-	0.8124	0.8124		
advent	100.459	-	0.8156	0.8156		
midmark	11.655	8.14E+10	0.8240	0.8240	Compiled with optimization turned on (O1) and linked against -lcii40-O1	Huge jump over initial start
sandmark	276.541	-	0.7855	0.9669		
advent	94.542	-	0.7675	0.9411		
midmark	10.931	8.65E+10	0.7728	0.9379	Compiled with optimization turned on (O2) and linked against -lcii40-O2	Not as big of a jump over O0 -> O1, but still significant
sandmark	273.209	-	0.7760	0.9880		
advent	94.191	-	0.7647	0.9963		
midmark	10.898	8.65E+10	0.7704	0.9970	Make read_instruction_line function inline	Not a signficiant change (inlining functions doesn't help much and read_instruction_line wasn't called that many times)
sandmark	263.603	-	0.7487	0.9648		
advent	93.103	-	0.7558	0.9884		
midmark	10.309	8.40E+10	0.7288	0.9460	Place UM_instruction functions in UM.c file (not using UM_instructions.o anymore); Made all UM_instruction functions "static inline"	Removes a layer / file of abstraction and inlines them (fairly insignificant)
sandmark	238.494	-	0.6774	0.9047		
advent	87.708	-	0.7120	0.9421		
midmark	9.689	7.95E+10	0.6850	0.9399	Removed excess Seq_length() and UM_get_seg() calls	These two functions are slow (using Hanson interface are slow) so minimize their use
sandmark	151.661	-	0.4308	0.6359		
advent	55.252	-	0.4486	0.6300		
midmark	6.096	4.72E+10	0.4310	0.6292	Declare Bitpack functions in UM.c and remove excess checks	Bitpack functions can be slow because they're filled with assert checks
sandmark	131.452	-	0.3734	0.8667		
advent	54.317	-	0.4410	0.9831		
midmark	5.264	3.61E+10	0.3721	0.8635	Declare Seg and related functions in UM.c (make static inline)	Removes layer of abstraction
sandmark	106.404	-	0.3022	0.8095		
advent	41.472	-	0.3367	0.7635		
midmark	4.167	3.16E+10	0.2946	0.7916	No longer use Seq to hold segments (use dynamic C-array)	Get rid of slow Hanson ADT, using faster C-array instead
sandmark	25.356	-	0.0720	0.2383		
advent	10.763	-	0.0874	0.2595		
midmark	1.019	7.11E+09	0.0720	0.2445	Remove all asserts except for when calling malloc()	Asserts are not that slow but the things they check can be (excessive calls for popular / slow functions every cycle); Made a HUGE improvement
sandmark	25.989	-	0.0738	1.0250		
advent	10.883	-	0.0884	1.0111		
midmark	1.022	7.04E+09	0.0723	1.0029	Made all functions (except for main 4 interface functions) static inline	Didn't make an improvement (actually made it slower)
sandmark	25.196	-	0.0716	0.9695		
advent	11.039	-	0.0896	1.0143	Remove p_counter variable in UM struct (move it to local variable in run_program())	Removes need to access element in UM_T memory

midmark	1.002	6.91E+09	0.0708	0.9804		
sandmark	23.019	-	0.0654	0.9136		
advent	9.749	-	0.0791	0.8831		
midmark	0.904	6.67E+09	0.0639	0.9022	Limit calling of Bitpack_getu to necessary cases	Removes excess calls to Bitpack functions (which are slow)
sandmark	23.095	-	0.0656	1.0033		
advent	10.049	-	0.0816	1.0308		
midmark	0.894	6.60E+09	0.0632	0.9889	Copied Stack functions in UM.c	Tried to trim down on slow Hanson ADTs
sandmark	23.506	-	0.0668	1.0178		
advent	9.979	-	0.0810	0.9930		
midmark	0.913	6.60E+09	0.0645	1.0213	Closed memory leak	Not for speed, just needed it
sandmark	22.368	-	0.0635	0.9516		
advent	9.109	-	0.0739	0.9128		
midmark	0.888	6.57E+09	0.0628	0.9726	Removed UM struct (made contents global)	Reduces the amount of pointer / structure passing needed to run the function (UM_T passed around a lot)
sandmark	12.796	-	0.0363	0.5721		
advent	4.898	-	0.0398	0.5377		
midmark	0.503	2.94E+09	0.0356	0.5664	Changed gcc compile script to remove -pg option	-pg unnecessarily calls mcount functions (used by gprof). Forgot to switch this off after we finished using gprof
sandmark	11.541	-	0.0328	0.9019		
advent	4.304	-	0.0349	0.8787		
midmark	0.465	2.82E+09	0.0329	0.9245	Changed words in Seg_T to flexible array; Tweaked expanding array parameters (made it expand more on each expand call)	Removed a level of interection on a commonly used array
sandmark	9.769	-	0.0277	0.8465		
advent	4.266	-	0.0346	0.9912		
midmark	0.389	2.82E+09	0.0275	0.8366	Removed unncessary libraries from compile script	Removed excess libaries that might slow down the code with extra weight