Benchmark	User time	Instructions	Relative to start	Relative to previous	Improvement
Midmark Advent Sandmark	4.25s 34.86s 107.24s	28.51 * 10^9 - -	1.000 1.000 1.000	1.000 1.000 1.000	No improvement (starting point)
Midmark Advent Sandmark	2.72s 22.88s 68.95s	23.30 * 10^9 - -	0.640 0.656 0.643	0.640 0.656 0.643	Compiled with optimization turned on and linked against -lcii-01 (gcc -01 and -lcii-01)
Midmark Advent Sandmark	2.49s 20.7s 63.78	22.02 * 10^9 - -	0.586 0.594 0.595	0.915 0.905 0.925	Compiled with optimization turned on and linked against -lcii-02 (gcc -02 and -lcii-02)
Midmark Advent Sandmark	1.58s 12.06 40.11s	92.92x10^8 - -	0.372 0.346 0.374	0.634 0.583 0.629	Bottleneck: calling bitpackgetu, especially to extract the opcode with every iteration of the driving loop Remedy: Changed all bitpackgetu calls to directly perform bitwise operations
Midmark Advent Sandmark	1.13s 7.49s 28.11s	61.94 *10^8	0.266 0.215 0.262	0.715 0.621 0.709	Bottleneck: Accessing instructions in mapped segments using UArray_at Remedy: Changed UArray_T representation segments of uint32_t pointers to C arrays of uint32_t values

Midmark Advent Sandmark	0.82s 5.44s 20.52s	51.95* 10^8	0.193 0.156 0.191	0.726 0.726 0.730	Bottleneck: Accessing mapped segments using Seq_get Remedy Changed Seq_T holding mapped memory segments into a C array of pointers to C arrays holding mapped segments
Midmark Advent Sandmark	0.68s 4.65s 17.51s	40.97 * 10^8	0.160 0.133 0.163	0.829 0.855 0.853	Not a bottleneck, more of a bug: Fixed memory allocation. Previously did not have to expand/ realloc arrays at any point, but were allocating memory in excess at beginning of the program. Changed allocation of memory to be dependent on the size of the original 0 segment, and reallocate if necessary. Removed all safety asserts at this step
Midmark Advent Sandmark	0.44s 3.79s 11.11s	31.84x10^8	0.104 0.108 0.104	0.647 0.815 0.634	Bottleneck: Creating segment IDs from unmapped IDs stored on a stack Remedy: Removed stack structure holding unmapped IDs and made it into a C array
Midmark Advent Sandmark	0.33s 3.06s 8.34s	24.13 x 10^8	0.078 0.089 0.078	0.75 0.807 0.75	Bottleneck: Function calls and compilation of included files Remedy: Put all code to handle

					construction, UM instructions, and memory deallocation into main
Midmark Advent Sandmark	0.32s 3.03s 8.16s	22.82 x 10^8	0.075 0.087 0.076	0.970 0.990 0.978	Bottleneck: Accessing struct member variables by dereferencing pointers Remedy: Removes structs for UM and UM memory and declared all member variables as local variables in main Not a significant change, but increased times ever so slightly so we kept it.
Midmark Advent Sandmark	0.35s 2.57s 8.68s	20.56x10^8	0.082 0.074 0.081	1.09 0.848 1.06	Bottleneck: “If” statements in main Remedy: Changed if branches into switch cases Note: this made midmark and sandmark benchmarks slower but got advent time lower than the reference