HW7 - UM Optimization

The following table enlists the phases of our optimization, and our rationales for optimisation decisions in terms of the bottlenecks we identified in our program's performance via the use of kcachegrind.

Phase	**Bottleneck**	**Midmark (sec)**	**Adventz (sec)**	**Sandmark (min:sec)**	**Instruction Fetches for Midmark**
Original	-	6.43	57.67	2:56.8	46,330,356,370
-01 optimized	-	5.85	55.97	2:23.88	31,944,441,611
-02 optimised	Bitpack_getu()	3.42	27.54	1:17.34	30,780,866,631
Bitpack functions duplicated + inlined	UArray_at()	2.04* 2.51	21.82	1:00.83	15, 270, 429,531
Made our own implementation of uarray, without asserts	Bitpack_getu() & Seq_get()	1.57	14.55	39.89	10,678,292,605
Made our own sequence	Non inlined functions seemed much slower	.57	6.26	14.31	5,187,942,910
Static inlining everything	UArray and Hanson's Mem functions	.34	2.86	8.49	3,325,910,576
Made our own stack	handle_instruction	.29	2.35	7.69	2, 981,883,165
Loop refactoring and removing unnecessary declarations structs and function calls (focusing on handle_instruction)	handle_instruction	.26	2.20	6.66	2,578,895,745

In the following table, the value on top represents time of the phase relative to the previous phase. The value below represents the time of the phase relative to the original unoptimised program.

Phase	Midmark	Advent	Sandmark
Original	-	-	-
	-	-	-
-01 Optimised	0.95533	0.97419	0.82021
	0.95533	0.97419	0.82021
-02 Optimised	0.57662	0.49279	0.53692
	0.55089	0.48009	0.44039
Bitpack functions duplicated + inlined	0.77027	0.80167	0.79136
	0.42432	0.38489	0.34850
Made our own implementation of uarray, without asserts	0.68421	0.66692	0.65952
	0.29032	0.25669	0.22984
Made our own sequence	0.48718	0.43128	0.35858
	0.14144	0.11070	0.08242
Static inlining everything	0.59649	0.53368	0.60735
	0.08437	0.05908	0.05006
Made our own stack	0.85294	0.75243	0.92439
	0.07196	0.04445	0.04627
Loop refactoring and removing unnecessary declarations structs and function calls (focusing on handle_instruction)	0.89656	0.81290	0.87117
	0.06452	0.03613	0.04031