Mason Pollack and Vladimir Hugec
Comp 40
HW7

Benchmark	Time (min:sec. ms)	Instructions (only found for midmark)	Rel to Start	Rel to Prev	Improvement
sandmark midmark	168.4 sec 6.81 sec	n/a 47.55E9	1.000 1.000	1.000 1.000	N/a, starting point
sandmark midmark	117.6 sec 4.59 sec	n/a 33.44E9	.698 .674	.698 .674	Compiled with optimization turned on and linked against -lcii-O1
sandmark midmark	109.2 sec 4.42 sec	n/a 33.48E9	.648 .649	.928 .963	Compiled with optimization turned on and linked against -lcii-O2
sandmark midmark	91.4 sec 3.71 sec	n/a 21.22E9	.543 .545	.837 .839	Made bitpack functions into static inline functions declared in the source file that uses the functions
sandmark midmark	67.48 sec 2.72 sec	n/a 17.47E9	.401 .399	.738 .733	Kept track of zero seg individually from other segments in order to reduce Seq_get calls to get the

					zero seg
sandmark midmark	59.28 sec 2.40 sec	n/a 13.60E9	.352 .352	.878 .882	Move function that increments pCounter (UM_data_increment_pCounter) into the file (UM_ALU.c) that calls it for every UM instruction executed. Declare function as static inline
sandmark midmark	35.8 sec 1.44	n/a 9.35E9	.213 .211	.604 .600	The individual zero seg that we are keeping track of was changed to be represented as an array instead of a Seq_T which allows for Seq_get to be used less and getting an instruction from the zeroSeg to be much quicker
sandmark midmark	27.20 .95	n/a	.172 .144	.760 .660	Put every function in 1 file and declared as static inline.

sandmark midmark					