Amoses Holton & Woody Butler
Comp 40 HW#7 Profiling
Notes taken 12/2/15-12/4/15

Benchmark	Time (sec)	Instructions	Rel. to Start	Rel. to Prev	Improvement
Big	102.31		1.00	-	Initial
Medium	35.28		1.00	-	Initial
Small	4.08	31,092,026,985	1.00	-	Initial
Big	94.03		0.9191	0.9191	-O1
Medium	32.9		0.933	0.933	-O1
Small	3.79	29,471,552,484	0.929	0.929	-O1
Big	94.29		0.9216	1.003	-O2
Medium	33.05		0.937	1.004	-O2
Small	3.80	29,472,972,013	0.931	1.002	-O2
Big	90.91		0.8886	0.9641	½ of functions from switch to array of function pointers
Medium	32.48		0.9206	0.9827	½ of functions from switch to array of function pointers
Small	3.65	29, 518, 845, 777	0.894	0.961	½ of functions from switch to array of function pointers
Big	82.55		0.807	0.908	5 More functions into func pointer array
Medium	28.93		0.820	0.891	5 More functions into func pointer array
Small	3.33	24,604,817,648	0.816	0.912	5 More functions into func pointer array

Big	49.52		0.484	0.5999	Changed to -O3, updated our flags to incorporate compile optimization. optimization.
Medium	17.27		0.4895	0.597	Changed to -O3, updated our flags to incorporate compile optimization.
Small	1.99	18, 879, 894, 515	0.488	0.598	Changed to -O3, updated our flags to incorporate compile optimization.
Big	32.69		0.320	0.660	Refactored bitpack_getu
Medium	11.49		0.325	0.665	Refactored bitpack_getu
Small	1.31	6,925,041,597	0.321	0.658	Refactored bitpack_getu
Big	16.56		0.161	0.507	Changed segment data structure for mapped segments.
Medium	6.20		0.178	0.540	Changed segment data structure for mapped segments.
Small	0.67	3,677,687,878	0.164	0.511	Changed segment data structure for mapped segments.
Big	12.12		0.118	0.732	Track local array of program instructions.
Medium	5.02		0.142	0.810	Track local array of program instructions.
Small	0.48	3,036,618,134	0.118	0.716	Track local array of program instructions.
Big	10.49		0.103	0.866	Grabbing 3 instructions per loop
Medium	4.85		0.137	0.966	Grabbing 3 instructions per loop
Small	0.43	3,036,616,675	0.105	0.896	Grabbing 3 instructions per loop

½ of functions from switch to array of function pointers
We took instructions 0-6 and put them into an array of function pointers, and then called those as opposed to using a switch-case statement to route those 7 instructions.

5 More functions into func pointer array
We moved instructions 8-12 into an array of function pointers (similar to the above step). Additionally, we shifted where we call bit_pack_getu so that it is not called unnecessarily as often.

Changed to -O3, updated our flags to incorporate compile optimization.
Updated our compiler optimization to 03, also noticed that we were compiling O0 in our flags and that was preventing us from using the previous O1 and O2 optimizations, but we changed that to use O3.

Refactored bitpack_getu
We inlined the bitpack_getu function ourselves and made modifications to what used to be each function call. We removed the asserts because we know that no error could throw off an assert within bitpack_getu that we hadn't caught with our um.c error checking.

Changed segment data structure for mapped segments.
We were using Hanson sequences to hold the segments (which were uint32_t arrays) but we changed that data structure to a 2D array.

Track local array of program instructions.
We were calling segment load every time we needed the next instruction, and to cut the middle man out we made a helper function within segment.c that returns a pointer to a segment. We called that function whenever a new program was loaded and used that pointer to iterate through the program.

Grabbing 3 instructions per loop
We began running the files under a run.sh script, which directed all output to /dev/null.