COMP40
Homework 7: Laboratory Notes
Lionel Oh (loh01) and Naoki Okada (nokada01)

Machine Model: Intel(R) Core(™) i7-6700 CPU@3.40GHz

Legend: *small* - `midmark` benchmark
mid - partial solution to `advent`
big - `sandmark` benchmark

No.	Benchmark	Time (User Mode)	Instructions	Rel to start	Rel to prev	Improvement /Change	Remarks
1	small	6.90s	5.3543×10^{10}	1.000	1.000	-	Starting Point (Without optimization flags)
	mid	60.63s	-	1.000	1.000		
	big	168.56s	-	1.000	1.000		
2	small	3.74s	3.4895×10^{10}	0.542	0.542	Compiled with optimization `-O1` flag turned on and linked against `-lcii40-O1`	-
	mid	33.64s	-	0.554	0.554		
	big	93.59s	-	0.555	0.555		
3	small	3.17s	3.3253×10^{10}	0.459	0.848	Compiled with optimization `-O2` flag turned on and linked against `-lcii40-O2`	Since this is faster, any improvements moving forward will be compiled with the the `-O2` flag
	mid	28.20s	-	0.465	0.838		
	big	81.39s	-	0.483	0.870		
4	small	2.87s	3.0400×10^{10}	0.416	0.905	Added a conditional within `run_prog` so that the variables `prog_seg` and `curr_length` are updated only when necessary (i.e. when the load program instruction is called)	This reduces the number of times `Seq_get` and `UArray_length` have to be called
	mid	25.75s	-	0.425	0.913		
	big	71.43s	-	0.424	0.878		
5	small	2.95s	3.1753×10^{10}	0.428	1.02	Took the checks for memory and register values out from the `ops_interface` functions, and instead put them within the `while` loop in `run_prog` -- **Undid change**	The change made it slower because the program was now checking all the variables within the `while` loop, when previously it was only checking the relevant variables within each function
	mid	26.57s	-	0.438	1.03		
	big	75.74s	-	0.449	1.06		
6	small	2.85s	3.0396×10^{10}	0.413	0.966	Modified the process in `init_prog` in which we determined how	Negligible improvement from change #4, but still a decrease in the
	mid	25.48s	-	0.420	0.959		

	big	72.47s	-	0.421	0.957	many `uint32_t` words there were in the file given. Instead of a `while` loop and using a counter variable, we utilized the C library's `fseek` function.	number of instructions nonetheless.
7	small	2.83s	3.0396×10^{10}	0.410	0.993	Put all functions in `mem_interface`, `ops_interface`, `io_dev` and `bitpack.c` in main	While the number of instructions seems to remain constant, apart from the anomalous increase of timing with mid, the overall timing does decrease slightly
	mid	25.57s	-	0.422	1.004		
	big	71.00s	-	0.430	0.980.		
8	small	1.70s	1.3380×10^{10}	0.246	0.601	Used `static inline` for all functions within `um.c`, aside from `main` and functions linked from `seq.h`	-
	mid	15.19s	-	0.251	0.594		
	big	42.63s	-	0.253	0.600		
9	small	1.70s	1.3326×10^{10}	0.246	0.000	Removed the `exit_condition` conditional; changed the `while` loop to always be true; `run_prog` returns when `prog_count` reaches the end of program	Not a significant decrease in time, but reduces the total number of instructions
	mid	14.93s	-	0.246	0.983		
	big	42.28s	-		0.992		
10	small	1.33s	1.0325×10^{10}	0.193	0.782	Changed the data structure used to represent the words within each segment, from a `UArray` to a dynamic C `uint32_t` array. As a corollary, also changed the `mem_seg` struct to include a `numwords` tracker that helps us keep track of the length of the C array	-
	mid	12.55s	-	0.207	0.841		
	big	33.25s	-	0.197	0.786		
11	small	0.93s	6.3407×10^{9}	0.135	0.699	Changed the data structure used to represent the registers, from a `UArray` to a static C `uint32_t` array	-
	mid	8.01s	-	0.132	0.638		
	big	23.32s	-	0.138	0.701		
12	small	0.38s	3.4349×10^{9}	0.055	0.409	Changed the data structure used to represent the memory,	-
	mid	2.71s	-	0.045	0.338		

	big	9.14s	-	0.054	0.392	from a `Seq` to a dynamic C array of `mem_seg`. As a corollary, also created a `memory` struct that includes the C array and a `memlength` tracker that helps us keep track of the length of the C array (i.e. number of segments)	
13	small	0.29s	3.0711×10^9	0.042	0.763	Changed the data structure used to keep track of the identifiers of unmapped identifiers, from a `Seq` to a dynamic C `uint32_t` array. As a corollary, also created an `unmapped_list` struct that includes the C array, a `lastindex` tracker that helps us keep track of the significant element of the C array, and a `listlength` variable for the size of the array	-
	mid	2.54s	-	0.042	0.937		
	big	7.57s	-	0.045	0.828		
14	small	0.29s	3.0765×10^9	0.042	0.000	Replaced `fseek` process used in change #6 with `stat` function outside of `init_prog`	Interestingly, the number of instructions increased, but timings became quicker
	mid	2.43s	-	0.040	0.957		
	big	7.30s	-	0.433	0.964		
15	small	0.27s	2.9905×10^9	0.039	0.931	Got rid of the `mem_seg` struct, and changed the data structure used to represent the memory, from a dynamic C array of `mem_seg` to a 2D dynamic C `uint32_t` array within the `memory` struct. The first element of the array is the old `mapped` boolean flag, the second element represents the old `numwords` tracker, and the rest of the array elements are the	-
	mid	2.13s	-	0.035	0.877		
	big	6.76s	-	0.401	0.926		

						words of the segment.	
15	small	0.25s	2.8971×10^9	0.036	0.926	Got rid of the `unmapped_list` struct, and moved all its variables into the `memory` struct.	-
	mid	2.01s	-	0.033	0.944		
	big	6.36s	-	0.038	0.941		
16	small	0.24s	2.8941×10^9	0.034	0.960	Compiled with optimization `-O3` flag turned on and linked against `-lcii40`	We will be using this compile flag in the final `Makefile`
	mid	1.94s	-	0.032	0.965		
	big	6.12s	-	0.036	0.962		