Ryan Beckwith (rbeckw02) and Victoria Chen (vchen05)
HW7: profiling
Date: 12/4/2020

Laboratory Notes

Benchmark	*Time*	*Instructions*	*Rel to start*	*Rel to prev*	*Improvement*
small[1]	5.33s	31.60×10^9	1.000	1.000	No improvement (starting point)
medium[2]	45.58s	—	1.000	1.000	
big[3]	133.48s	—	1.000	1.000	
small	3.76s	25.40×10^9	0.705	0.705	Compiled with optimization turned on and linked against `-lcii-O1`
medium	31.66s	—	0.695	0.695	
big	95.25s	—	0.714	0.714	
small	3.78s	25.40×10^9	0.709	1.005	Compiled with optimization turned on and linked against `-lcii-O2`
medium	31.49s	—	0.691	0.995	
big	94.46s	—	0.708	0.992	
small	3.70s	24.75×10^9	0.694	0.979	Removed intermediate data storage struct for the Load Value instruction utilized by the Unpacker interface, directly utilized bitpacking interface instead.
medium	30.90s	—	0.677	0.981	
big	93.10s	—	0.697	0.986	
small	3.12s	23.79×10^9	0.585	0.843	Removed intermediate data storage struct for all other instructions utilized by the Unpacker interface, directly utilized bitpacking interface instead.
medium	27.19s	—	0.597	0.880	
big	78.16s	—	0.586	0.840	
small	3.27s	23.42×10^9	0.614	1.048	Removed the unpacker function that retrieves the opcode by directly utilizing the bitpacking interface instead - removed in the next step.
medium	27.29s	—	0.598	1.003	
big	78.59s	—	0.588	1.005	
small	2.82s	20.05×10^9	0.529	0.862	Retrieved the opcode directly via a single right-shift operation instead of utilizing the bitpacking interface.
medium	24.73s	—	0.542	0.906	
big	70.81s	—	0.530	0.901	
small	2.49s	17.44×10^9	0.467	0.883	Unpacked the register A index for the Load Value instruction with a bitwise AND and a right-shift operation, and the corresponding value with a bitwise AND operation.
medium	21.49s	—	0.471	0.869	
big	62.18s	—	0.466	0.878	
small	1.89s	11.19×10^9	0.355	0.759	Unpacked register indices A, B, and C for general 3-register instructions with bitwise AND and right-shift operations.
medium	15.22s	—	0.334	0.708	
big	47.60s	—	0.357	0.766	
small	1.15s	7.53×10^9	0.216	0.608	Retrieved segment 0 and saved it in a local variable, exploiting temporal locality for accessing program instructions.
medium	8.38s	—	0.184	0.551	
big	29.32s	—	0.220	0.616	

[1] midmark.um
[2] advent.umz
[3] sandmark.umz

small medium big	1.08s 6.97s 25.93s	7.19×10^9 — —	0.203 0.153 0.194	0.939 0.831 0.884	Replaced a wrapper function for extracting a word from a segment with a direct call to `UArray_at` to remove unnecessary function-calling overhead.
small medium big	0.97s 6.77s 24.87s	7.01×10^9 — —	0.182 0.149 0.186	0.898 0.971 0.959	Replaced the `Mem_update_word` function with its equivalent instructions directly in the instruction executor to remove unnecessary function-calling overhead.
small medium big	0.99s 6.74s 25.39s	6.89×10^9 — —	0.186 0.148 0.190	1.021 0.996 1.020	Replaced the `Mem_get_word` function with its equivalent instructions directly in the instruction executor to remove unnecessary function-calling overhead.
small medium big	0.87s 6.55s 22.72s	6.04×10^9 — —	0.163 0.143 0.170	0.879 0.972 0.895	Prevented unnecessary deallocation of segments when the Unmap Segment operation is called, allowing for reuse of old segments.
small medium big	0.87s 6.55s 22.99s	6.04×10^9 — —	0.163 0.143 0.172	1.000 1.000 1.012	Removed unnecessary retrieval of segment 0 when initializing program instructions from the input .um file.
small medium big	0.83s 6.50s 22.60s	5.74×10^9 — —	0.156 0.143 0.169	0.954 0.992 0.983	Removed unnecessary retrieval of segments when initializing each element in a new segment to 0.
small medium big	0.85s 5.76s 21.78s	5.79×10^9 — —	0.159 0.120 0.163	1.024 0.886 0.963	Moved the opcode switch statement from the `execute_cases` function to the `read_instructions` function to remove unnecessary function-calling overhead.
small medium big	0.55s 4.05s 14.71s	3.25×10^9 — —	0.103 0.089 0.110	0.647 0.703 0.675	Replaced Hanson's `UArray` with a new lightweight data structure, `SArray`, that stores all segments in a 1D array (later removed).
small medium big	0.55s 4.09s 14.59s	3.26×10^9 — —	0.103 0.090 0.109	1.000 1.010 0.992	Replaced wrapper `Mem_T` struct directly with its contents as local variables to reduce unnecessary pointer indirection.
small medium big	0.34s 2.80s 8.56s	2.47×10^9 — —	0.064 0.061 0.064	0.618 0.685 0.587	Replaced our `Seq_T` of `SArray`s (representing main memory) with a single dynamic C array, storing segment information in a helper dynamic C array of structs.
small medium big	0.29s 2.53s 7.50s	2.39×10^9 — —	0.054 0.056 0.056	0.853 0.904 0.876	Replaced our `Seq_T` that stores the deleted addresses with a single dynamic C array to reduce pointer chasing and unnecessary assertions.