Names: Tam A. Luong and Sartrapat Saengcharoentrakul
Usernames: tluong04 and ssaeng01
Assignment 7

LABNOTES

Files chosen for benchmarks:
+ Small benchmark: midmark.um
+ Big benchmark: advent.umz (with "n", "take bolt", and "quit" as input)

Valgrind Run	Benchmark	Time	Instructions	Relative to start	Relative to previous	Improvement
	Big	66.313	-	1.000	1.000	No improvement (starting point)
	Small	8.176	8.0 x 10^7	1.000	1.000	
	Big	50.039	-	0.755	0.755	Compiled with Optimization turned on and linked against -lcii-01
	Small	6.157	8.0 x 10^7	0.753	0.753	
	Big	44.416	-	0.670	0.888	Compiled with Optimization turned on and linked against -lcii-02
	Small	5.484	8.0 x 10^7	0.671	0.891	
1	Big	39.543	-	0.596	0.891	Eliminate bitpack_getu() for opcode 0-12
	Small	4.976	53,521,922,710	0.609	0.907	
	Big	37.555	-	0.567	0.950	Eliminate bitpack_getu() for opcode 13
	Small	4.661	53,521,922,710	0.571	0.937	
2	Big	33.805	-	0.510	0.900	Eliminate the use of progCounter functions
	Small	4.241	36,393,384,141	0.519	0.910	
3	Big	29.184	-	0.440	0.863	Changed the segment implementation from UArray to C-Array
	Small	3.586	31,043,724,017	0.439	0.846	
4	Big	24.006	-	0.362	0.823	Change the register implementation from
	Small	3.048	26,460,977,863	0.373	0.850	

						Uarray to C-Array
5	Big	22.349		0.337	0.931	Replace register_get() and register_put() with direct c-array access
	Small	2.839	23,443,685,796	0.347	0.931	
6	Big	17.085	-	0.258	0.764	Replace segment_isEmpty() with direct access
	Small	2.170	18,313,839,202	0.265	0.764	
7	Big	10.130	-	0.153	0.593	Replace segment_length() with direct access
	Small	1.339	8,897,436,767	0.164	0.617	
8	Big	9.772	-	0.147	0.965	Replace segment_get with direct memory access
	Small	1.262	8,609,926,252	0.154	0.942	
9	Big	5.673	-	0.086	0.581	Store segment 0 separately as an array
	Small	0.742	5,677,025,661	0.091	0.588	
10	Big	4.035	-	0.061	0.711	Replace Hanson's uarray implementation to carray implementation for main memory
	Small	0.440	3,548,948,276	0.054	0.593	
11	Big	3.803	-	0.0573	0.943	Replace Hanson stack implementation with Carray stack implementation. This is used in the unmapped ID stack in the memory.
	Small	0.392	3,171,475,895	0.0480	0.890	

Notes on Valgrind runs:

First Valgrind run:
callgrind.out.11602
First Improvement:
We noticed that a lot of time is spent in Bitpack_getu,

Second Valgrind run:
callgrind.out.4590
We noticed that Uarray_at takes up a lot of of time, with 26.88%. So we decide to change the segment implementation from Uarray to CArray

Third Valgrind run:
callgrind.out.8410
We noticed that Uarray_at still takes a lot of time so we decided to change the register implementation from Uarray to Carray too.

Forth Valgrind run:
callgrind.out.9929
We noticed that register_get() and register_put() take a significant time to run. So we decided to replace these functions with direct memory access/

Fifth Valgrind run:
callgrind.out.11023
We noticed that Segment_isEmpty takes a lot of time. Hence we decided to replace the function with direct memory access from the memory

Sixth Valgrind run:
callgrind.out.11575
We noticed that Segment_length takes a lot of time.

Seventh Valgrind run:
callgrind.out.5773
We noticed that segment_get still takes up a big portion of the running time. We will try to remove segment_get as well.

Eighth Valgrind run:
callgrind.out.6734
We noticed that Uarray_get() takes up a lot of time. Thus, we are removing segment 0 from the Uarray so that the program can call it less.

Ninth Valgrind run:
callgrind.out.3257
We noticed that the main memory still use Hanson's Uarray implementation and that takes a lot of time. We decided to replace it with Carray.

Tenth Valgrind run:
callgrind.out.4175
We noticed that everything has been squashed down to minimal (both instruction-wise and visually on

kcachegrind) to give space to just our main functions such as execute, segment_load, segment_store, etc.

However, we also noticed that we still use stack from Hanson and that takes up a significant portion of time. We decide to replace it with our own implemented stack.

Eleventh Valgrind run:

callgrind.out.8102

We noticed that there is no more Hanson codes in the final version. All of the functions shows up are functions defined as part of the UM machine.