Stage	Benchmark	Time (sec)	Instructions	Relative to Start	Relative to Previous 'Kept' Change	Change	Notes	Kept?
0	midmark	4.303	26,043,531,732	1.000	1.000	Starting Point	N/A	Yes
	sandmark	102.362		1.000	1.000			
	advent (given partial soln.)	36.609		1.000	1.000			
1	midmark	2.522	20,011,264,355	0.586	0.586	compile with -O1 and link with -lcii40-O1	Immediate improvement; midmark efficiency improved by over 6 million instructions	Yes
	sandmark	62.158		0.607	0.607			
	advent	21.352		0.583	0.583			
2	midmark	2.470	19,828,558,590	0.574	0.979	compile with -O2 and link with -lcii40-O2	Strictly better than -O1 by a slim margin; opted to keep this over -O1	Yes
	sandmark	61.853		0.604	0.995			
	advent	21.344		0.583	1.000			
3	midmark	2.468	18,907,048,077	0.574	0.999	Only calculate necessary Um_registers for the current operation	Fixed inefficiency where operations that needed < 3 registers were still retrieving all 3 registers	Yes
	sandmark	61.651		0.602	0.997			
	advent	20.998		0.574	0.984			
4	midmark	1.905	16,708,830,094	0.443	0.772	Move all nested variable declarations outside loops	Surprisingly easy and significant improvement; definitely worth doing in the future	Yes
	sandmark	48.253		0.471	0.783			
	advent	16.439		0.449	0.783			
5	midmark	0.709	5,138,772,072	0.165	0.372	Copy bitpack functions to local files, make them inline	INCREDIBLE IMPROVEMENT. Bitpack functions went from ~60% of the program to < 20%	Yes
	sandmark	17.817		0.174	0.369			
	advent	6.123		0.167	0.372			
6	midmark	0.586	4,412,684,562	0.136	0.827	Make all operation functions inline	Inspired by previous change to static inline, led to worthwhile improvements	Yes
	sandmark	15.024		0.147	0.843			
	advent	5.056		0.138	0.826			
7	midmark	0.546	3,243,828,893	0.127	0.932	Replace chain of if-else statements for operations with switch	Tiny improvement in hotspot led to very noticeable change	Yes
	sandmark	13.976		0.137	0.930			
	advent	4.598		0.126	0.909			
8	midmark	0.545	3,243,828,846	0.127	0.998	Replace pointer to UM struct with global variable	Remove layer of abstraction, small improvements	Yes
	sandmark	13.840		0.135	0.990			
	advent	4.385		0.120	0.954			
9	midmark	0.540	3,243,828,846	0.125	0.991	Move all functions to one file	advent is a little slower, possibly from noise; overall faster, so we keep this change	Yes
	sandmark	13.797		0.135	0.997			
	advent	4.388		0.120	1.001			
10	midmark	0.565	3,243,828,846	0.131	1.046	Replace enum values with #define - slower across the board; change reverted	Slower across the board; reverted.	No
	sandmark	15.264		0.149	1.106			
	advent	4.478		0.122	1.021			
11	midmark	0.535	3,137,441,961	0.124	0.991	Remove redudant Seq_get calls for when loading a program	Programs load faster, especially advent	Yes
	sandmark	13.218		0.129	0.958			
	advent	3.904		0.107	0.890			
12	midmark	0.339	2,233,927,271	0.079	0.634	Replace Hanson Sequence representation of segments with custom dynamic array	INCREDIBLE IMPROVEMENT - nearly halved runtime from previous change	Yes
	sandmark	8.400		0.082	0.635			
	advent	2.570		0.070	0.658			
13	midmark	0.313	2,235,343,026	0.073	0.923	Replace sizeof statements with constant precalculated values	Removed redundant calculations from program hotspot	Yes
	sandmark	7.787		0.076	0.927			
	advent	2.333		0.064	0.908			
14	midmark	0.309	2,235,343,026	0.072	0.987	Remove redundant include statements	Minimal but worth keeping	Yes
	sandmark	7.781		0.076	0.999			
	advent	2.329		0.064	0.998			
15	midmark	0.298	2,160,977,360	0.069	0.964	Replace Hanson Sequence for unmmaped ids with custom-written inline dynamic array	Faster with a custom structure that removes unnecessary checks	Yes
	sandmark	7.762		0.076	0.998			
	advent	2.321		0.063	0.997			
16	midmark	0.294	2,159,738,744	0.068	0.987	Replace Hanson Sequence used for reading in instructions with dynamic array	Completely removed all dependency on Hanson!	Yes
	sandmark	7.651		0.075	0.986			
	advent	2.294		0.063	0.988			
17	midmark	0.292	2,164,049,504	0.068	0.993	Remove all redundant arithmetic calculations	Small performance boost; strange that there are slightly more instructions, but since it is faster, we keep it	Yes
	sandmark	7.581		0.074	0.991			
	advent	2.239		0.061	0.976			
18	midmark	0.270	2,066,711,250	0.063	0.925	Change the unsigned word from uint_64 to uint_32 in the Bitpack functions	Reduces the operations needed for bitshift and improves the overall performance further	Yes
	sandmark	6.826		0.067	0.900			
	advent	2.153		0.059	0.962			

All tests were run with:

```
vm-hw05{kgao03}104: grep "model name" /proc/cpuinfo
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
model name      : Intel(R) Xeon(R) Silver 4214Y CPU @ 2.20GHz
```