Risc0 Segmentation Test Results

Reza Hasanzadeh
Wholesum Network
rezahsnz@proton.me

July 9th, 2024

1 Introduction

Risc0 is a general-purpose zkVM that enables for arbitrary Rust programs to become verifiable. Given an input Rust program, Risc0 compiles it into a RISC-V binary ELF and feeds into its zk-STARK circuitry producing a verifiable execution trace represented by a *guest* program. This guest program, when run, produces proofs that can be used by third-parties to check if the execution was indeed honest. In order to have infinitely-sized programs, Risc0 divides a guest program into *segments* with segment size capped at 2^{24} *cycles*[1]. If a zkVM program does not fit into ≈16m cycles, it needs to be partitioned into segments and proved one by one. Segment size can be customized to lower limits too with the acceptable range starting at 2^{13} cycles. Along with execution steps, each segment size has particular memory requirements shown on Table 1(a comprehensive benchmarking is available here).

Segment size	**Minimum required memory**
cycles	≈ *amount*
$2^{13} \approx 8k$	64 MB
$2^{14} \approx 16k$	128 MB
$2^{15} \approx 32k$	256 MB
$2^{16} \approx 64k$	512 MB
$2^{17} \approx 128k$	1 GB
$2^{18} \approx 256k$	2 GB
$2^{19} \approx 512k$	4 GB
$2^{20} \approx 1M$	8 GB
$2^{21} \approx 2M$	16 GB
$2^{22} \approx 4M$	32 GB
$2^{23} \approx 8M$	64 GB
$2^{24} \approx 16M$	128 GB

Table 1: Memory requirements per segment size

Real-world zkVM programs are often big in size. Next section provides a detailed record of inflating and proving a typical guest program. It measures performance and storage requirements of various segment size limits.

[1] A cycle is roughly equivalent to a RISC-V operation.

2 Segmentation test results

In order to emulate a big guest program, a large vector of random u32 numbers were written to the guest memory in each round. While there are more sophisticated inflation methods, this method is easy[2] and it can be used as a baseline for further experiments. To segment the guest program, we limited its *po2-size-limit* in each round resulting in multiple independent segments per limit size. These segments were then saved as binary blobs to disk for later individual proving. There are 8 rounds of segmentation and proving in total where each round limits the segment size from 2^{15} up to 2^{24}. To measure performance of individual segment proving, a subset of segment blobs were drawn at random(uniform distribution), proved, and the maximum values were recorded. The test computer is a *Lenovo Thinkpad e450(Intel Core i7 5500U(2C 4T), 16GB RAM)* laptop running an *Ubuntu 18.04* dockerized-Risc0-1.0.1 environment.

Rounds

Round 1: up to *1m* cycles
User cycles: *872,543*
Inflation method: write 11k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	169	26	10	5	3	1	-	-	-	-
Total cycles	5.5M	1.7M	1.3M	1.2M	1.1M	$\approx 1M$[a]	-	-	-	-
Induced inflation	5.281x	1.625x	1.250x	1.125x	1.063x	1.000x	-	-	-	-
Prove time	14s	28s	56s	120s	262s	541s	-	-	-	-
Sample size	25%	50%	70%	80%	100%	100%	-	-	-	-
Blob size	19kb	37kb	52kb	88kb	158kb	298kb	-	-	-	-

[a] Exact value: 1,048,576

Table 2: Segmentation of a 1M cycles guest

Round 2: up to *2.5m* cycles
User cycles: *1,820,543*
Inflation method: write 23k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	333	53	20	9	5	3	1	-	-	-
Total cycles	11m	3.5m	2.6m	2.4m	2.2m	2.2m	$\approx 2.1m$[a]	-	-	-
Induced inflation	5.203x	1.656x	1.250x	1.125x	1.063x	1.031x	1.000x	-	-	-
Prove time	14s	29s	58s	123s	262s	540s	-	-	-	-
Sample size	15%	40%	50%	50%	70%	100%	-	-	-	-
Blob size	19kb	35kb	52kb	88kb	158kb	299kb	581kb	-	-	-

[a] Exact value: 2,097,152

Table 3: Segmentation of a 2.5M cycles guest

[2] RISC architecture tends to have lower number of instructions, so a cycle representing a load instruction, for example, is not inferior to the one representing a mathematical operation.

Round 3: up to *5m* cycles
User cycles: *3,637,543*
Inflation method: write 46k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	647	105	40	18	9	4	2	1	-	-
Total cycles	21.2m	6.9m	5.2m	4.6m	4.3m	4.2m	4.2m	$\approx 4.2m$[a]	-	-
Induced inflation	5.055x	1.641x	1.234x	1.094x	1.031x	1.000x	1.000x	1.000x	-	-
Prove time	13s	28s	57s	120s	249s	536s	-	-	-	-
Sample size	2.5%	15%	20%	25%	30%	75%	-	-	-	-
Blob size	19kb	35kb	52kb	88kb	158kb	301kb	581kb	1.1mb	-	-

[a] Exact value: 4,194,304

Table 4: Segmentation of a 5M cycles guest

Round 4: up to *10m* cycles
User cycles: *7,271,543*
Inflation method: write 92k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	1,278	209	78	35	17	8	4	2	1	-
Total cycles	41.9m	13.7m	10.2m	9.2m	8.7m	8.4m	8.4m	8.4m	$\approx 8.4m$[a]	-
Induced inflation	4.992x	1.633x	1.219x	1.094x	1.031x	1.000x	1.000x	1.000x	1.000x	-
Prove time	14s	29s	57s	119s	248s	539s	-	-	-	-
Sample size	3%	10%	15%	15%	20%	30%	-	-	-	-
Blob size	19kb	37kb	52kb	88kb	158kb	299kb	581kb	1.2mb	2.2mb	-

[a] Exact value: 8,388,608

Table 5: Segmentation of a 10M cycles guest

Round 5: up to *20m* cycles
User cycles: *15,013,543*
Inflation method: write 190k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	2,616	429	161	71	34	17	9	5	3	1
Total cycles	85.7m	28.1m	21m	18.6m	17.8m	17.3m	17m	16.9m	16.8m	$\approx 16.8m$[a]
Induced inflation	5.109x	1.676x	1.254x	1.109x	1.063x	1.031x	1.016x	1.008x	1.004x	1.000x
Prove time	14s	29s	59s	121s	251s	539s	-	-	-	-
Sample size	1%	5%	10%	10%	10%	15%	-	-	-	-
Blob size	19kb	37kb	52kb	88kb	158kb	299kb	581kb	1.2mb	2.3mb	4.5m

[a] Exact value: 16,777,216

Table 6: Segmentation of a 20M cycles guest

Round 6: up to *45m* cycles
User cycles: *32,390,116*
Inflation method: write 400k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	7,187	1,019	379	168	80	39	20	10	5	3
Total cycles	235.5m	66.8m	49.7m	44m	41.7m	40.9m	40.1m	39.8m	41.9m	$\approx 41.9m$[a]
Induced inflation	5.615x	1.592x	1.184x	1.050x	0.994x	0.975x	0.956x	0.950x	1.000x	1.000x
Prove time	15s	30s	58s	119s	245s	541s	-	-	-	-
Sample size	0.1%	2.5%	5%	5%	5%	10%	-	-	-	-
Blob size	20kb	36kb	67kb	133kb	265kb	528kb	1.1mb	1.7mb	2.8mb	5.2mb

[a] Exact value: 41,943,040

Table 7: Segmentation of a 45M cycles guest

Round 7: up to *90m* cycles
User cycles: *65,563,120*
Inflation method: write 800k u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	16,808	2,174	801	355	168	82	41	20	10	5
Total cycles	551m	142.5m	105m	92.9m	87.8m	85.5m	84.4m	83.9m	83.9m	$\approx 83.9m$[a]
Induced inflation	6.566x	1.698x	1.252x	1.107x	1.047x	1.019x	1.006x	1.000x	1.000x	1.000x
Prove time	14s	28s	58s	119s	249s	534s[b]	-	-	-	-
Sample size	0.1%	2.5%	5%	5%	5%	3%	-	-	-	-
Blob size	19kb	38kb	67kb	133kb	265kb	529kb	1.1mb	2mb	4.1mb	6.4mb

[a] Exact value: 83,886,080
[b] low value is due to small sample size

Table 8: Segmentation of a 90M cycles guest

Round 8: up to *180m* cycles
User cycles: *131,908,988*
Inflation method: write 1.6m u32 numbers to the guest
The results are displayed on the following table:

Segment size	2^{15}	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}	2^{21}	2^{22}	2^{23}	2^{24}
# Segments	36,046	4,483	1,643	727	344	167	83	41	21	11
Total cycles	1.18b	293.8m	215.4m	190.6m	180.1m	175.1m	173m	172m	172m	$\approx 172m$[a]
Induced inflation	6.869x	1.708x	1.252x	1.108x	1.047x	1.018x	1.006x	1.000x	1.000x	1.000x
Prove time	15s	29s	66s	124s	257s	544s	-	-	-	-
Sample size	0.1%	2.5%	5%	5%	5%	2%	-	-	-	-
Blob size	19kb	35kb	67kb	133kb	265kb	529kb	1.1mb	2mb	4.1mb	7.4mb

[a] Exact value: 171,966,464

Table 9: Segmentation of a 180M cycles guest

Conclusion

This note investigated a scenario where a Risc0 guest program gets inflated with random inputs. 8 rounds of tests were conduced on segmentation and individual segment proving with a focus on systems whose memory are capped at 8GB. The results point to sweet spots on 2^{18} and 2^{19} segment size limits where parallelized proving setups could be employed to prove prohibitively large programs quickly with acceptable overhead and inflation in total cycles.