CS401

Computer Architectures

2022-2023 Spring

Term Project

Phase II

Ege Zorlutuna

Muhammed Orhun Gale

3.1 Round Operations:

In this part of the project, a round operation function “roundOperation” has been created. This function takes the addresses of rkey and s value and updates t value on the data segment of the MIPS implementation.

With the given test vectors to check the correctness of the program, roundOperation function updates the t value as intended output. Given test vectors, the expected output and the output of the roundOperation function can be seen below.

3.1.1 Test Vector:

Given test vectors:

s: .word 0xd82c07cd, 0xc2094cbd, 0x6baa9441, 0x42485e3f

rkey: .word 0x82e2e670, 0x67a9c37d, 0xc8a7063b, 0x4da5e71f

Expected t state:

t = [0x2892750e, 0x949a0d1f, 0x70523edc, 0xc6933381]

Output of the roundOperation in the MIPS implementation:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10040000	0x10040400	0x10040800	0x10040c00	0x6573552f	0x6d2f7372	0x7568726f	0x6f442f6e
0x10010020	0x6f6c6e77	0x2f736461	0x30345343	0x72505f31	0x63656a6f	0x61742f74	0x73656c62	0x7461642e
0x10010040	0x00000000	0x2892750e	0x949a0d1f	0x70523edc	0xc6933381	0xd82c07cd	0xc2094cbd	0x6baa9441
0x10010060	0x42485e3f	0x82e2e670	0x67a9c37d	0xc8a7063b	0x4da5e71f	0x36637830	0x33363336	0x202c3561
0x10010080	0x38667830	0x63376337	0x202c3438	0x65657830	0x37373737	0x202c3939	0x36667830	0x62376237
0x100100a0	0x202c6438	0x66667830	0x32663266	0x202c6430	0x36647830	0x62366236	0x202c6462	0x65647830
0x100100c0	0x66366636	0x202c3162	0x31397830	0x35633563	0x202c3435	0x30367830	0x30333033	0x202c3035

3.1.2 Cache Performance:

In order to test the cache performance and increase the lookup table access performance by increasing the hit rate, the MIPS implementation of phase 1 and the roundOperation was put on “roundOperationTest.asm” file. The program ran on MARS with the required cache configurations given on Table 1 and answers to the following questions were found.

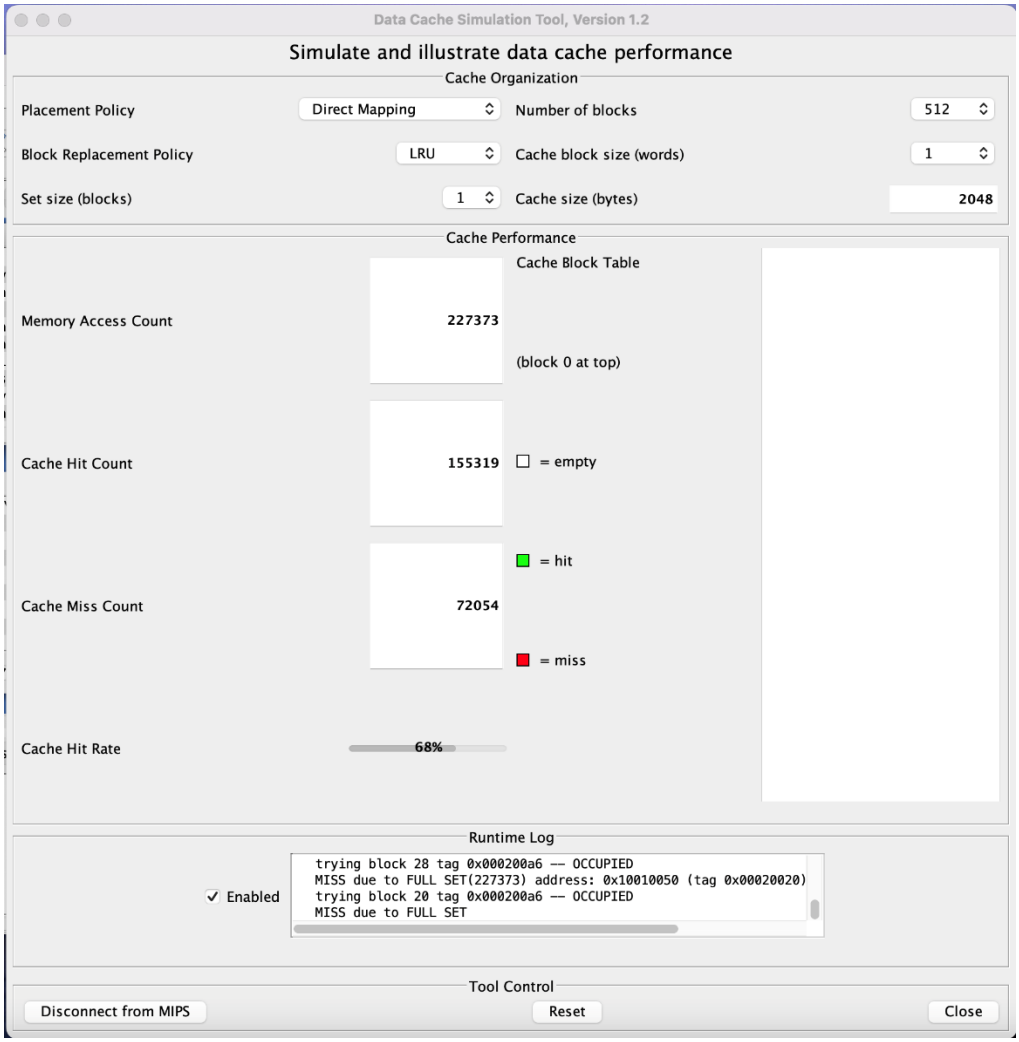
Block Size	No. of Blocks	Cache Size	Hit Rate
4 B	512	2048 B	?
8 B	128	1024 B	?
16 B	32	512 B	?
32 B	16	512 B	?
64 B	8	512 B	?

Table 1: Cache Configuration

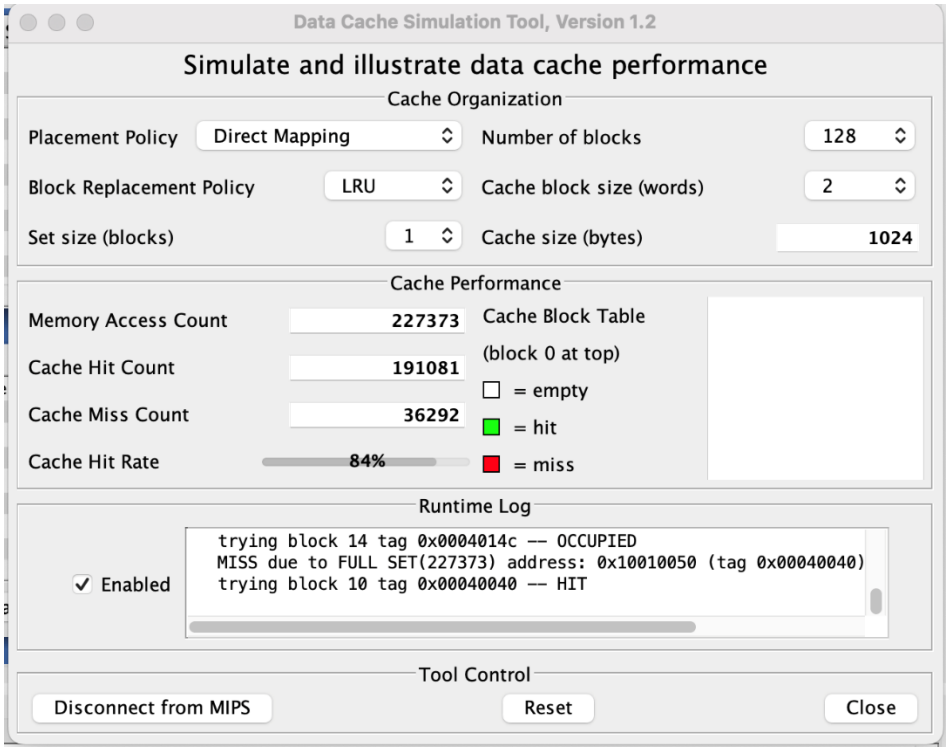
1) **Find the hit rates for each cache configuration.**

Results of the experiments and hit rates found:

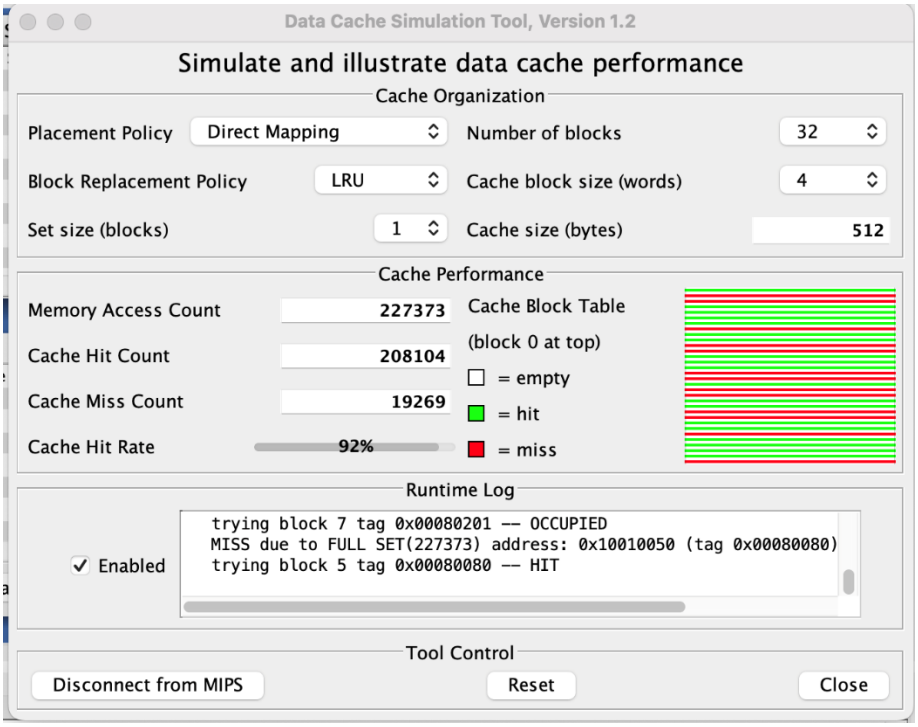
1- Block Size: 4B (1 word), No. of Blocks: 512, Cache Size: 2048B, Hit Rate: 68%



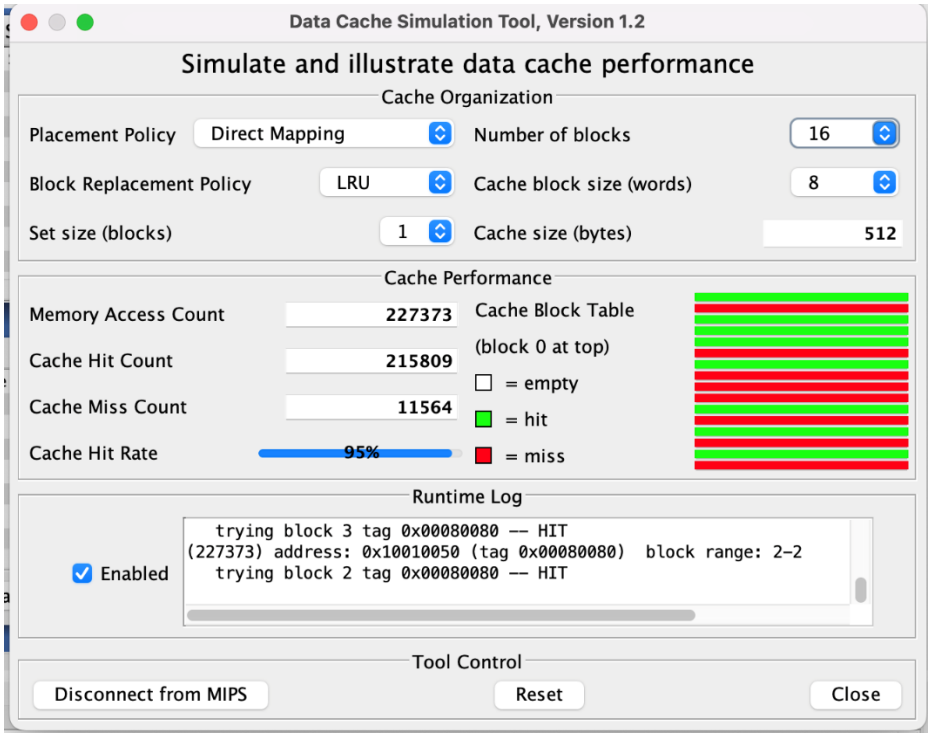
2- Block Size: 8B (2 words), No. of Blocks: 128, Cache Size: 1024B, Hit Rate: 84%



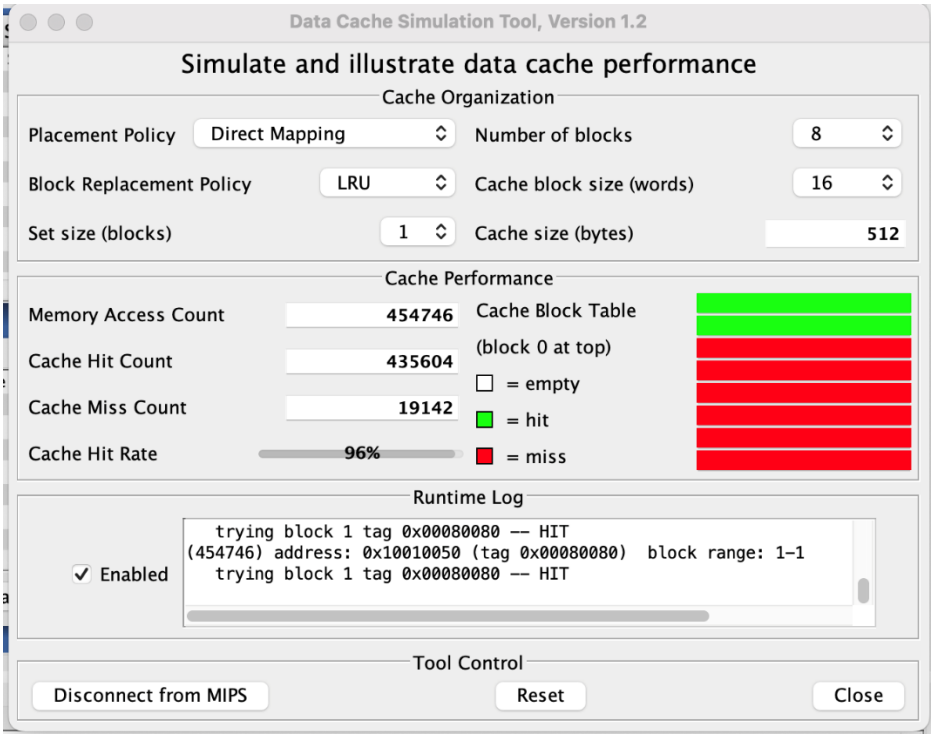
3- Block Size: 16B (4 words), No. of Blocks: 32, Cache Size: 512B, Hit Rate: 92%



4- Block Size: 32B (8 words), No. of Blocks: 16, Cache Size: 512B, Hit Rate: 95%



5- Block Size: 64B (16 words), No. of Blocks: 8, Cache Size: 512B, Hit Rate: 96%



2) **Which cache configuration yields the best performance?**

   Block Size: 64B (16 words), No. of Blocks: 8, Cache Size: 512B, Hit Rate: 96%

3) **Does the largest cache always give the best result? Why or why not?**

   No, even though increasing the cache size can improve performance up to a point, the largest cache size does not always yield in the best result in performance. Furthermore, a too large cache can affect performance contribution of the cache adversely. Having a larger cache can result on increased hit and miss latencies since they store more data to search among the one that would be used on the run. In addition, a large cache with too small block size cannot exploit spatial locality well enough and lowers the performance contribution, which can be also observed on the conducted experiment. As a result, it can be stated that the cache characteristics for the best performance depends on several factors such as cache size, block size, replacement policy, latency, etc.

3.2 Key Schedule:

In this part of the project, “updaterkey” and “keyschedule” functions have been implemented and can be found on “cs401_TPphase2_morhun_egezorlutuna.asm” together with the “roundOperation” function. updaterkey function takes the addresses of the rkey (should be initialized with the value of silent key on the data segment of MIPS implementation) and the address of the proper rcon value. As a result, this function updates the given rkey properly. keychedule function executes updatekey function 8 times and after each execution of updatekey function, rkey value updated properly. (For this phase, only the execution of updaterkey function and finding proper rkey values was enough, but roundOperation function was also called additionally. After each key is generated, roundOperation called with updated key and remaining state and returns updated state after each call.)

With the given silent key to check the correctness of the program, keyschedule function updates the rkey value 8 times, and finds the intended rkey values on each update. Given secret key, the expected result of rkey after 8 iterations and the resulting rkey from implemented keyschedule function can be seen below (resulting rkey from keyschedule function will be the 8^{th} rkey since it updates rkey value 8 times properly).

Given secret key:

key = [0x6920e299, 0xa5202a6d, 0x656e6368, 0x69746f2a]

Expected value of rkey after 8th update:

round key 8 = [0xc0194bc5, 0xd005973f, 0x39cfc711, 0xbf9c0a7c]

Result of rkey after 8 updates using updatekey and keyschedue functions on the MIPS implementation provided:

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x10040000	0x10040400	0x10040800	0x10040c00	0x6573552f	0x6d2f7372	0x7568726f	0x6f442f6e
0x10010020	0x6f6c6e77	0x2f736461	0x30345343	0x72505f31	0x63656a6f	0x61742f74	0x73656c62	0x7461642e
0x10010040	0x00000000	0x45d81dcf	0xa3c7445b	0xc819380d	0xe4f270c8	0xd82c07cd	0xc2094cbd	0x6baa9441
0x10010060	0x42485e3f	0xc0194bc5	0xd005973f	0x39cfc711	0xbf9c0a7c	0x00000080	0x00000040	0x00000020
0x10010080	0x00000010	0x00000008	0x00000004	0x00000002	0x00000001	0x36637830	0x33363336	0x202c3561
0x100100a0	0x38667830	0x63376337	0x202c3438	0x65657830	0x37373737	0x202c3939	0x36667830	0x62376237
0x100100c0	0x202c6438	0x66667830	0x32663266	0x202c6430	0x36647830	0x62366236	0x202c6462	0x65647830