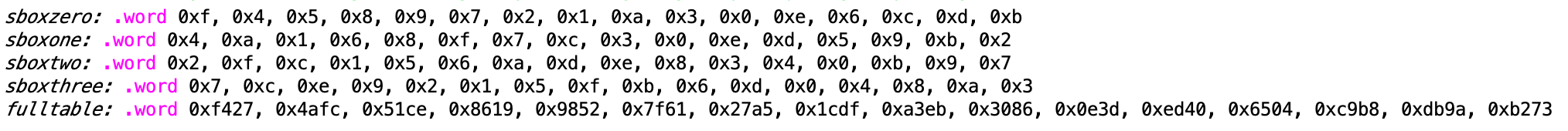
CS401 Term Project Phase IV Report

ilkergul 26352 & armantorikoglu 25005

**Memory/Cache access performance**

After completing the single table and multiple table methods, we wanted to compare cache miss and cache rate results with two different cache arrangements. Unfortunately, we did not create a cache that is 8 KB. Since we do not have that many words that we need to store, we made small caches and compared the results via the methods mentioned above.

Initially, let’s look at how we store the table.

****

In the multi-table approach, we accessed the corresponding word of the respected sbox table. First, however, we merged all the tables into one table and accessed the related word in the table, and processed the corresponding hexadecimal value.

Now, let’s look at the results. Firstly, we chose the number of blocks as two and cache block size as 16 since the table consists of 16 words. We used a 2-way Set Associative placement policy and got the results demonstrated below.

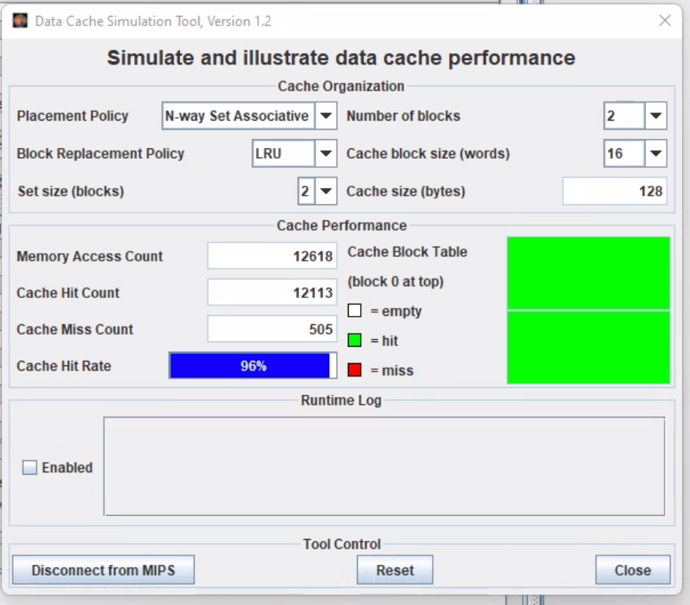
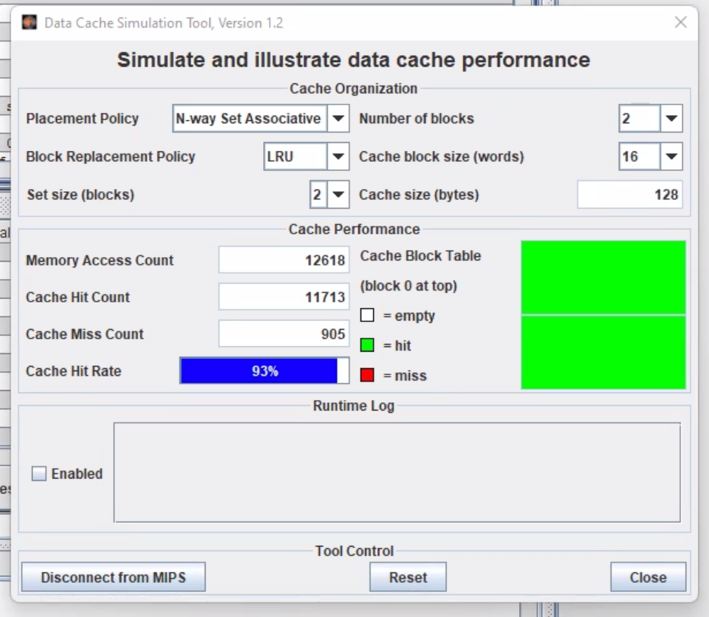


Figure 1: Multi Table Approach Figure 2: Single Table Approach

As we have expected, we got a better cache hit rate in the single table approach since we try to access only one table and its corresponding words, while in the multi-table method, we try to access four different tables.

Secondly, we chose the number of blocks as four and cache block size as 16 since the table consists of 16 words. We used the Direct Mapping placement policy and got the results demonstrated below.

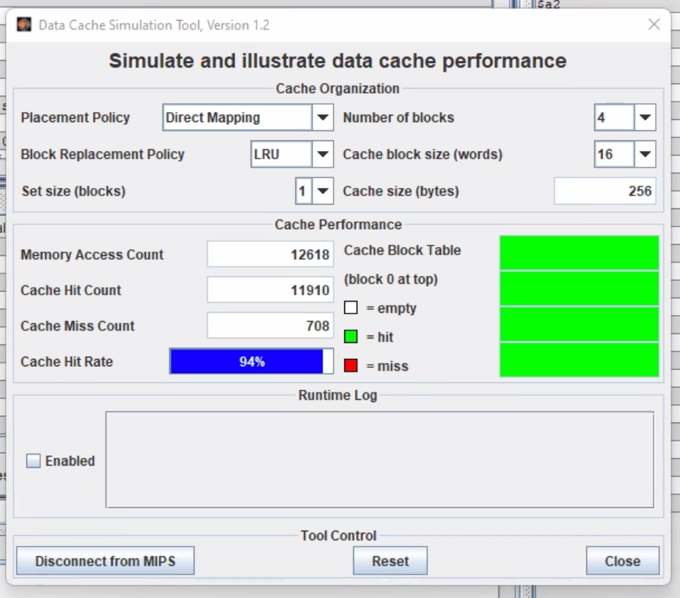
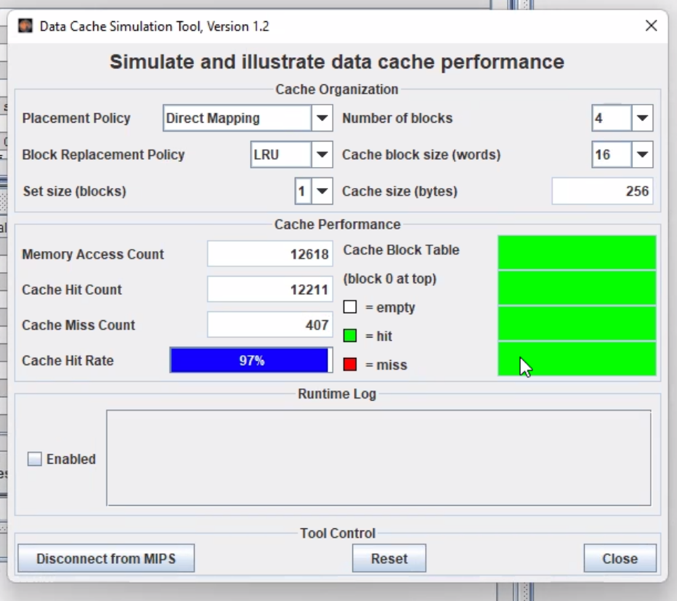
 

Figure 3: Multi Table Approach Figure 4: Single Table Approach

We received a better cache hit rate in the single table strategy, as expected, because we only tried to access one table and its accompanying words, whereas, in the multi-table method, we tried to access four tables. As a result of using direct mapping, some blocks may experience a cache miss/cache hit dilemma continuously during the execution, as we mentioned in the lectures.

**Final Result**

After implementing the necessary functions in phase II, which is tested and demonstrated below, we added the Encryption/Decryption functions and tested them with given sample vectors (figure 6). Finally, we used those functions to input a plaintext, encrypt it, then decrypt it to acquire the same result as is shown in figure 7. At the end, all of the inputs from the previous phases remain, and the user first has to input a number to calculate its F(X), then several numbers to test the W function, and another number to encrypt and decrypt to achieve the same result.

Text

Description automatically generated with low confidence

Figure 5: Phase II Demo

 Figure 6: Phase III Test

Graphical user interface, text, application, email

Description automatically generated Figure 7: Final Result