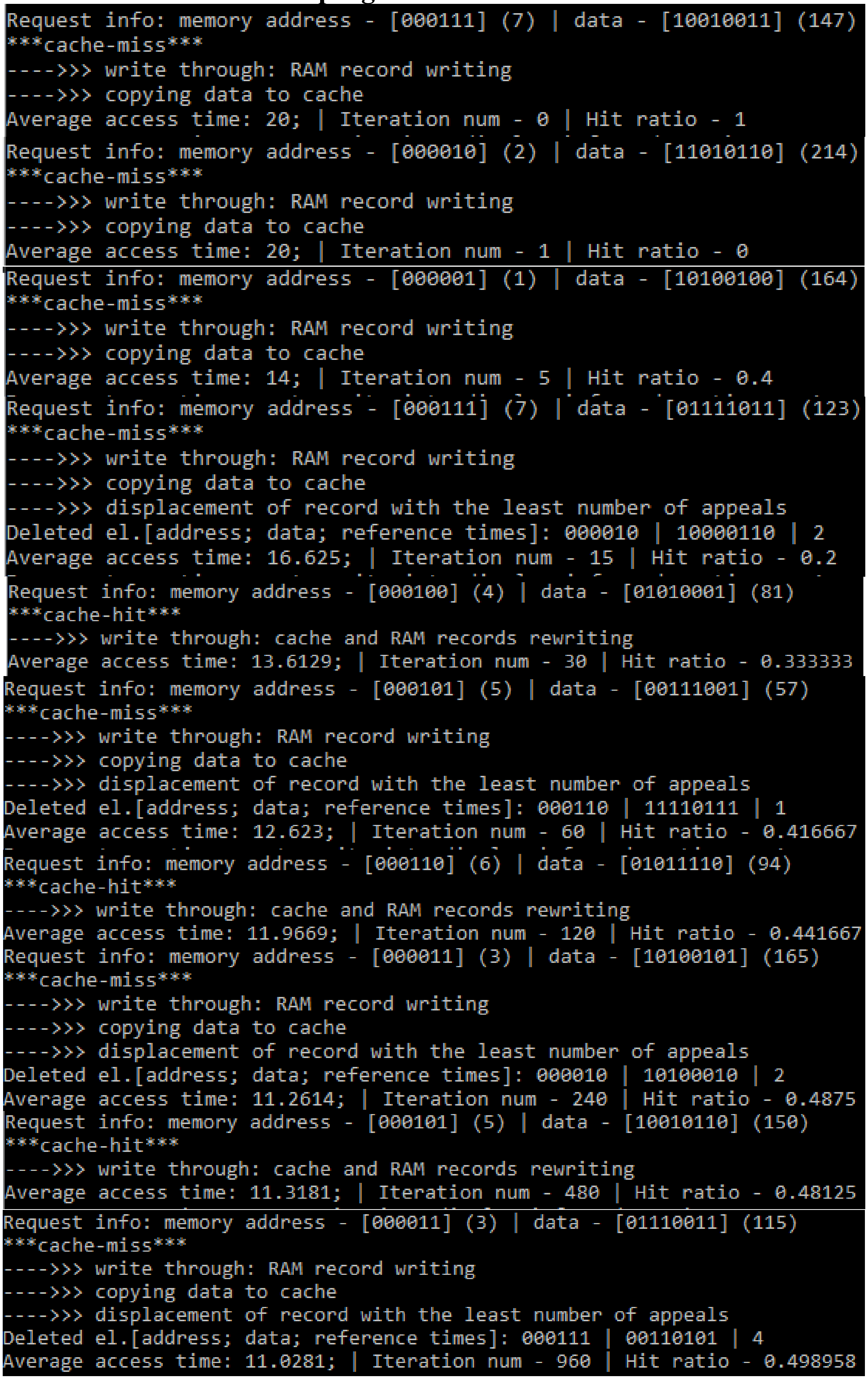
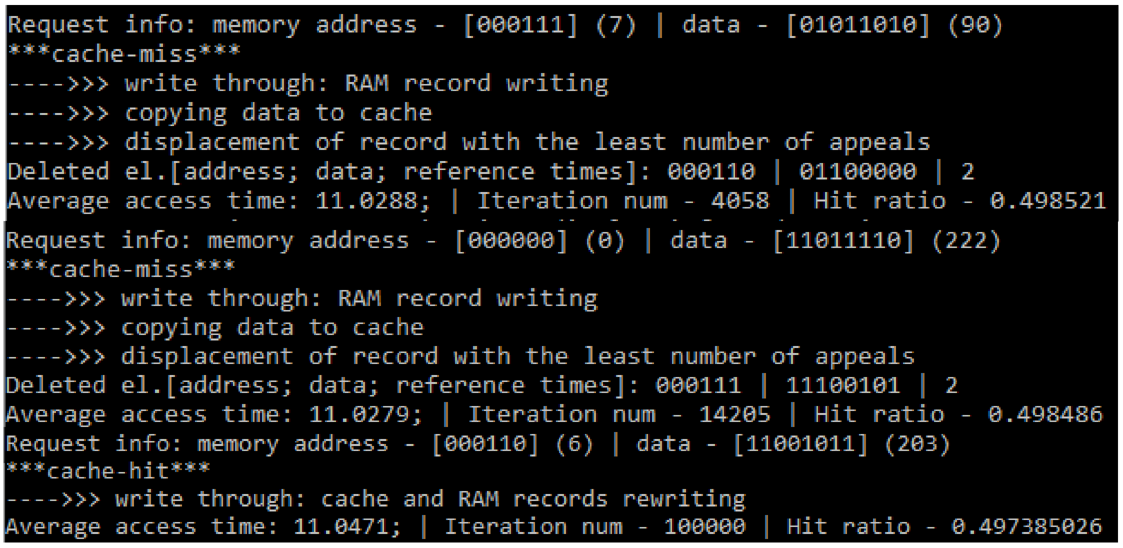
Application launch

```
       Main info:
|- memory bit size: 6 -|
|- data bit size: 8 -|
|- RAM table size: 8 -|
|- cache table size: 4 -|
|- hit ratio - 1 hit_n - 0 miss_n - 0 -|
|- T RAM - 20 T cache - 2 -|

Request info: memory address - [000111]
(7) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record
writing
---->>> copying data to cache
Average access time: 20; | Iteration num
- 0
Press c to continue, q to exit, i to
display info and continue...^
t to show tables and coninue...

t

---------------------------cache table|
| index| cache address |       data |
|     7|         000111|     10010011|
0

----------------------RAM table|
|   RAM address |        data |
|              7|     10010011|
Request info: memory address - [000010]
(2) | data - [11010110] (214)
***cache-miss***
---->>> write through: RAM record
writing
---->>> copying data to cache
Average access time: 20; | Iteration num
- 1
Press c to continue, q to exit, i to
display info and continue...^
t to show tables and coninue...

t

---------------------------cache table|
| index| cache address |       data |
|     7|         000111|     10010011|
0
|     2|         000010|     11010110|
1

----------------------RAM table|
|   RAM address |        data |
|              2|     11010110|
|              7|     10010011|
Request info: memory address - [000111]
(7) | data - [01111001] (121)
***cache-hit***
---->>> write through: cache and RAM
records rewriting
Average access time: 14; | Iteration num
- 2
Press c to continue, q to exit, i to
display info and continue...^
t to show tables and coninue...

t

---------------------------cache table|
| index| cache address |       data |
|     7|         000111|     01111001|
2
|     2|         000010|     11010110|
1

----------------------RAM table|
|   RAM address |        data |
|              2|     11010110|
|              7|     01111001|
Request info: memory address - [000000]
(0) | data - [01000110] (70)
***cache-miss***
---->>> write through: RAM record
writing
---->>> copying data to cache
Average access time: 15.5; | Iteration
num - 3
Press c to continue, q to exit, i to
display info and continue...^
t to show tables and coninue...

t

---------------------------cache table|
| index| cache address |       data |
|     7|         000111|     01111001|
2
|     2|         000010|     11010110|
1
|     0|         000000|     01000110|
3

----------------------RAM table|
|   RAM address |        data |
|              0|     01000110|
|              2|     11010110|
|              7|     01111001|
Request info: memory address - [000010]
(2) | data - [10000111] (135)
***cache-hit***
---->>> write through: cache and RAM
records rewriting
Average access time: 12.8; | Iteration
num - 4
Press c to continue, q to exit, i to
display info and continue...^
t to show tables and coninue...
```

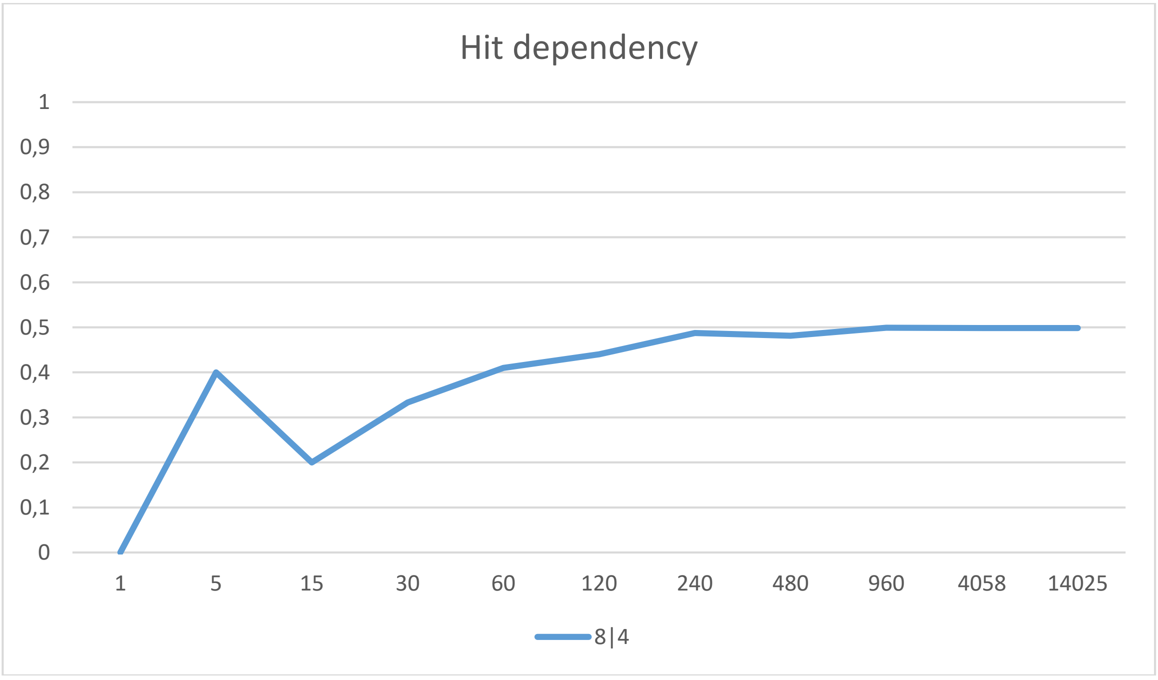
Checking the average access time and hit probability at different segments of the program execution

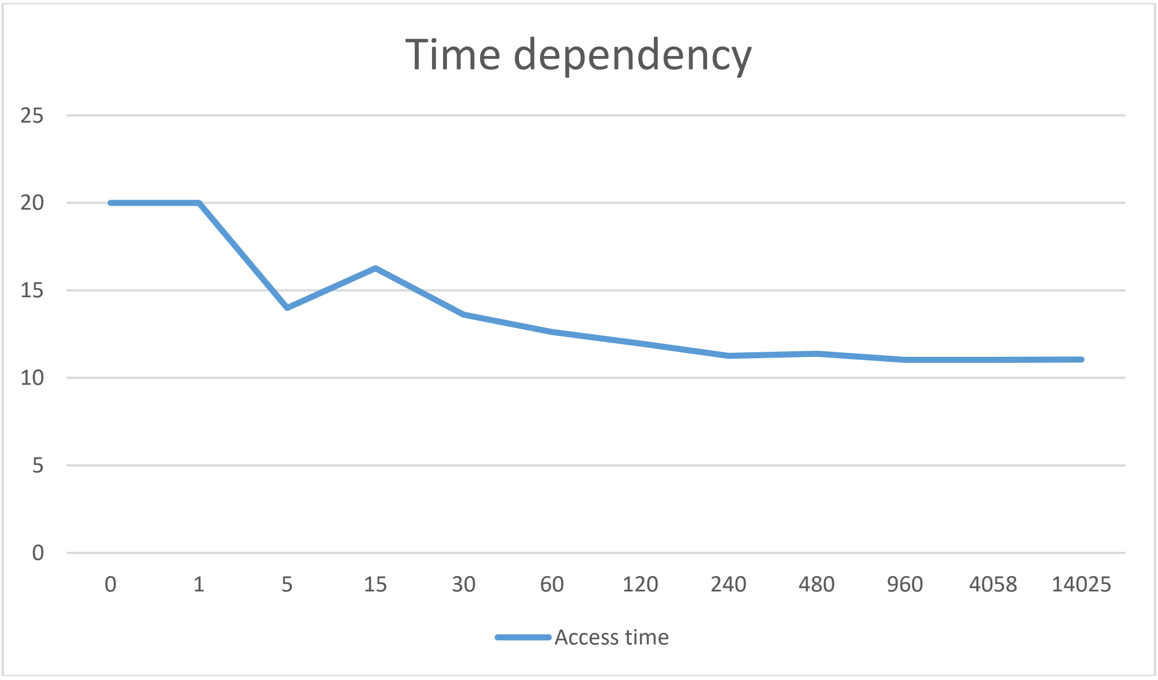


```
Request info: memory address - [000111] (7) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 0 | Hit ratio - 1
Request info: memory address - [000010] (2) | data - [11010110] (214)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 1 | Hit ratio - 0
Request info: memory address - [000001] (1) | data - [10100100] (164)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 14; | Iteration num - 5 | Hit ratio - 0.4
Request info: memory address - [000111] (7) | data - [01111011] (123)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000010 | 10000110 | 2
Average access time: 16.625; | Iteration num - 15 | Hit ratio - 0.2
Request info: memory address - [000100] (4) | data - [01010001] (81)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 13.6129; | Iteration num - 30 | Hit ratio - 0.333333
Request info: memory address - [000101] (5) | data - [00111001] (57)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000110 | 11110111 | 1
Average access time: 12.623; | Iteration num - 60 | Hit ratio - 0.416667
Request info: memory address - [000110] (6) | data - [01011110] (94)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.9669; | Iteration num - 120 | Hit ratio - 0.441667
Request info: memory address - [000011] (3) | data - [10100101] (165)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000010 | 10100010 | 2
Average access time: 11.2614; | Iteration num - 240 | Hit ratio - 0.4875
Request info: memory address - [000101] (5) | data - [10010110] (150)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.3181; | Iteration num - 480 | Hit ratio - 0.48125
Request info: memory address - [000011] (3) | data - [01110011] (115)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000111 | 00110101 | 4
Average access time: 11.0281; | Iteration num - 960 | Hit ratio - 0.498958
```

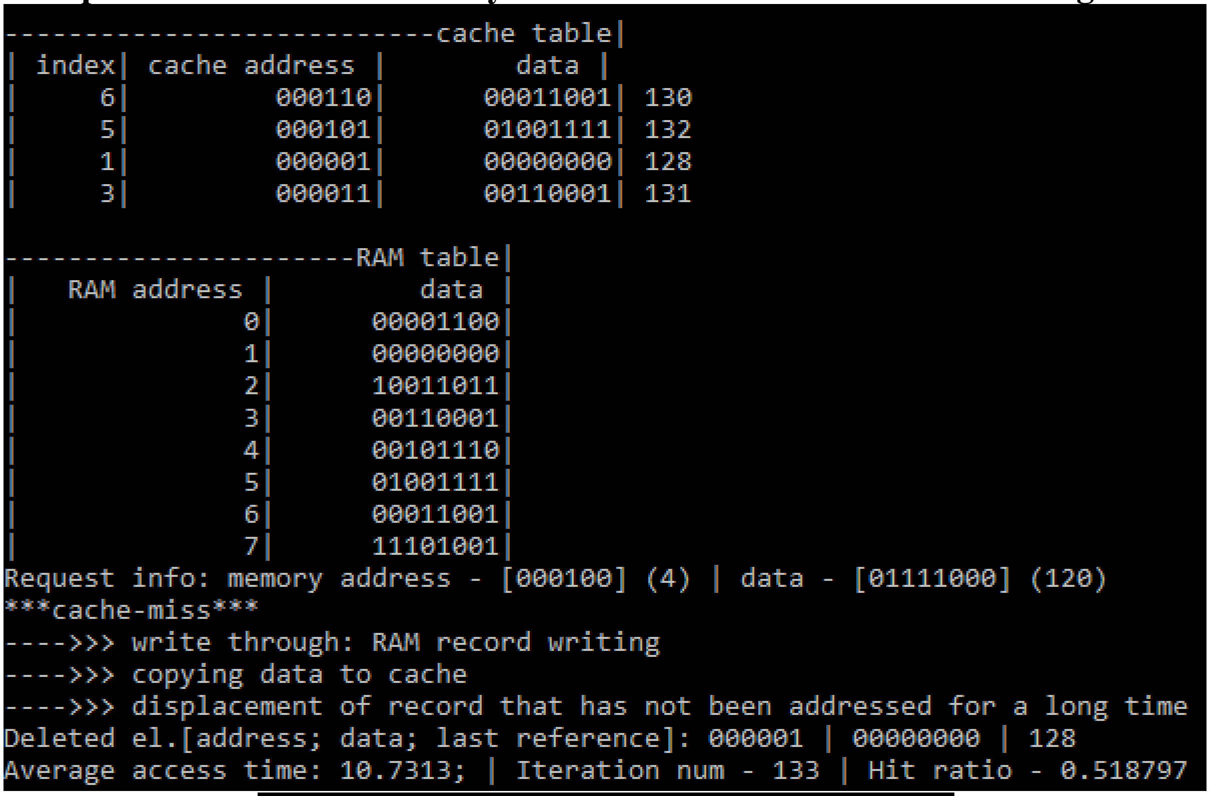


```
Request info: memory address - [000111] (7) | data - [01011010] (90)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000110 | 01100000 | 2
Average access time: 11.0288; | Iteration num - 4058 | Hit ratio - 0.498521
Request info: memory address - [000000] (0) | data - [11011110] (222)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000111 | 11100101 | 2
Average access time: 11.0279; | Iteration num - 14205 | Hit ratio - 0.498486
Request info: memory address - [000110] (6) | data - [11001011] (203)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.0471; | Iteration num - 100000 | Hit ratio - 0.497385026
```

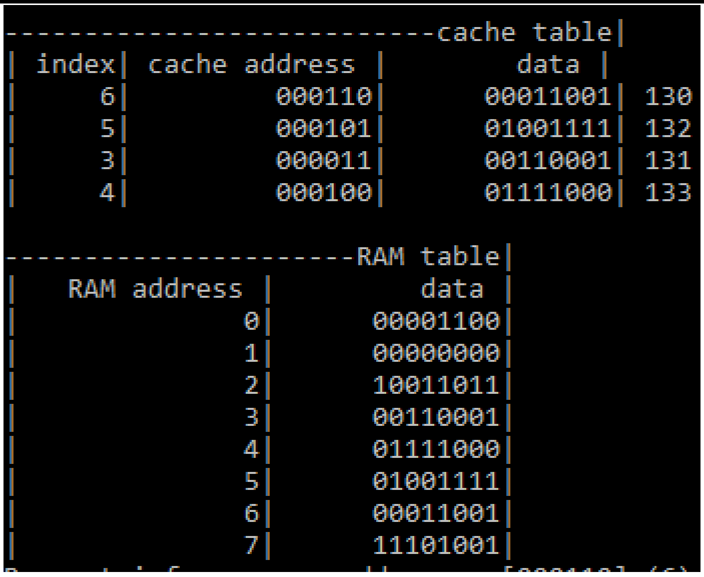




Replacement of a cache entry that has not been accessed for a long time

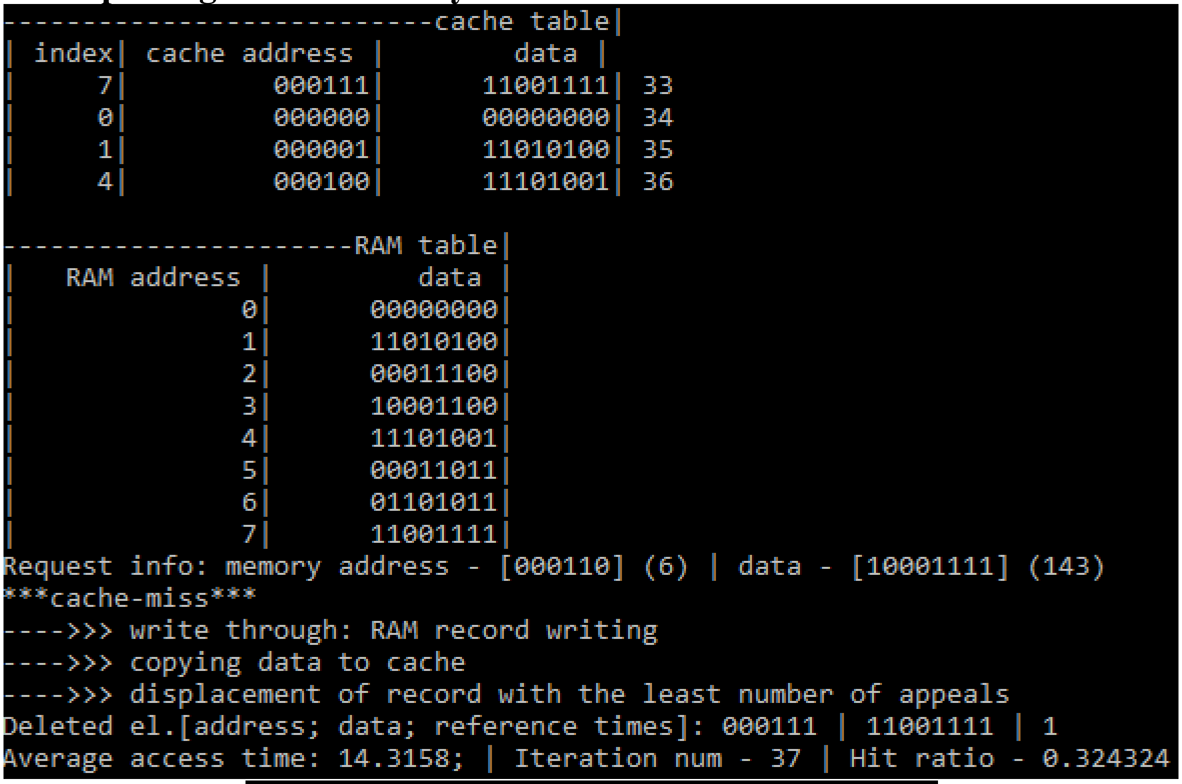


```
---------------------------cache table|
| index| cache address |        data |
|     6|         000110|      00011001| 130
|     5|         000101|      01001111| 132
|     1|         000001|      00000000| 128
|     3|         000011|      00110001| 131

----------------------RAM table|
|   RAM address |         data |
|              0|      00001100|
|              1|      00000000|
|              2|      10011011|
|              3|      00110001|
|              4|      00101110|
|              5|      01001111|
|              6|      00011001|
|              7|      11101001|
Request info: memory address - [000100] (4) | data - [01111000] (120)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 000001 | 00000000 | 128
Average access time: 10.7313; | Iteration num - 133 | Hit ratio - 0.518797
```

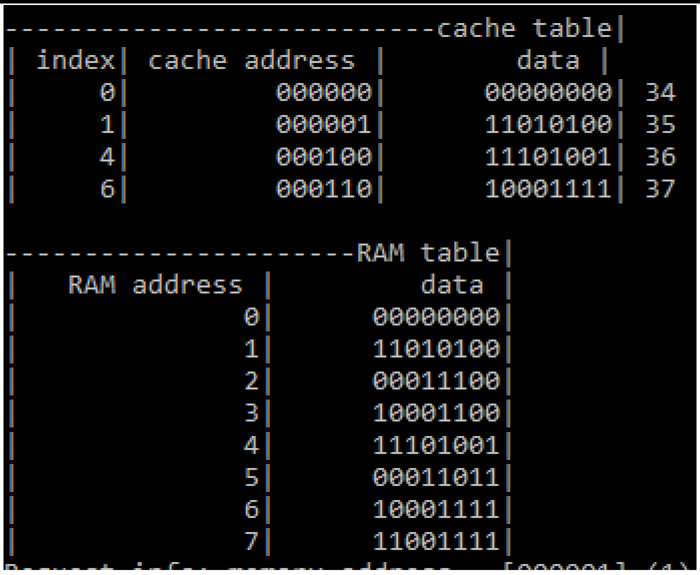


```
---------------------------cache table|
| index| cache address |        data |
|     6|         000110|      00011001| 130
|     5|         000101|      01001111| 132
|     3|         000011|      00110001| 131
|     4|         000100|      01111000| 133

----------------------RAM table|
|   RAM address |         data |
|              0|      00001100|
|              1|      00000000|
|              2|      10011011|
|              3|      00110001|
|              4|      01111000|
|              5|      01001111|
|              6|      00011001|
|              7|      11101001|
```

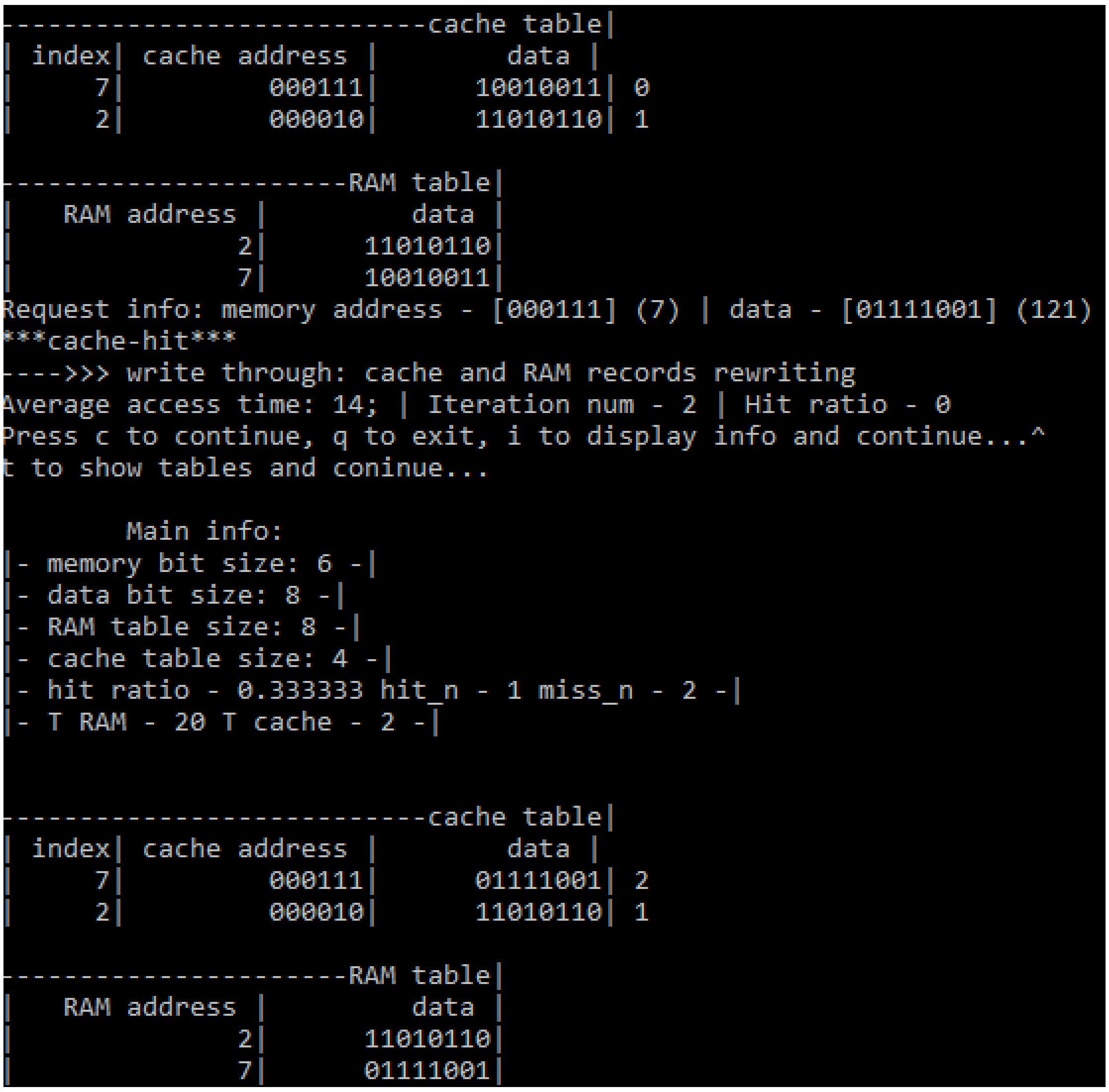
Replacing the cache entry that was called the least number of times



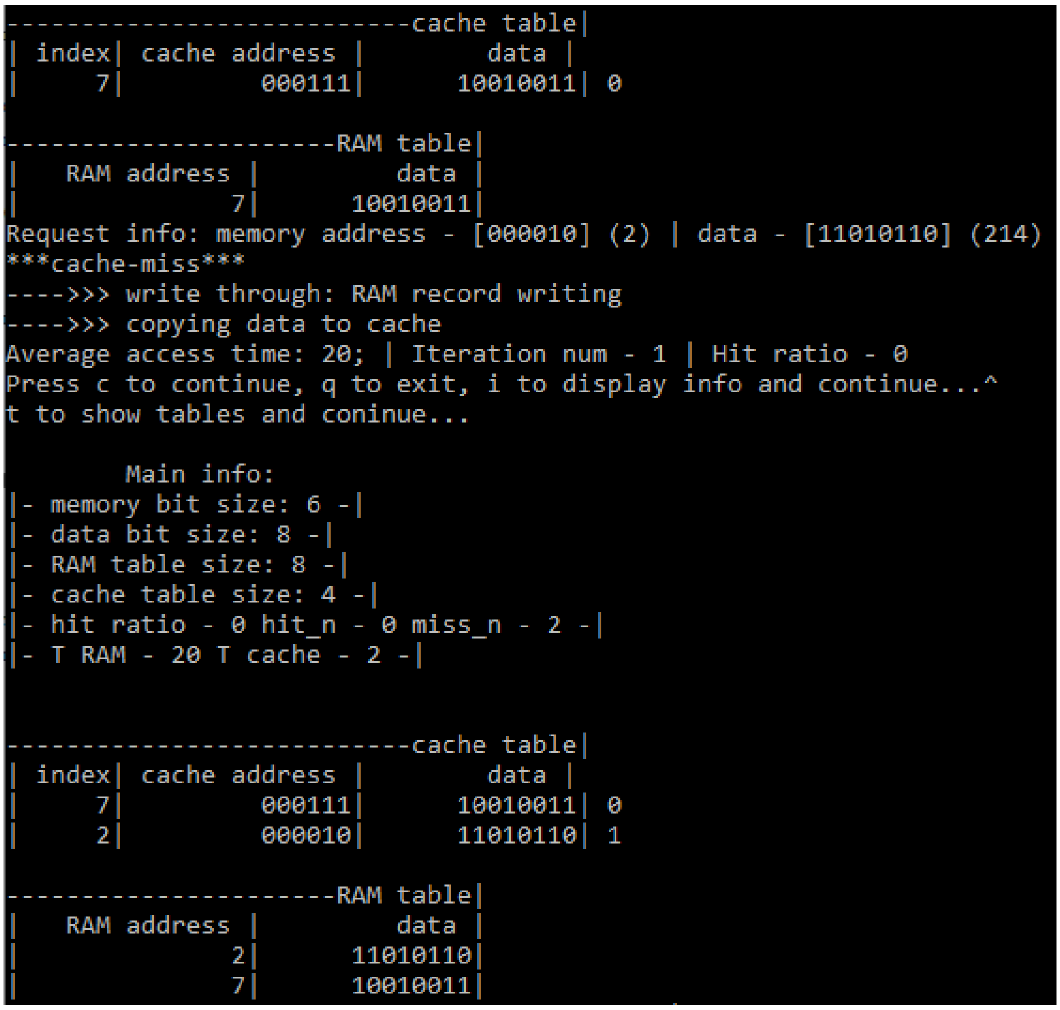
```
---------------------------cache table|
| index| cache address |        data |
|     7|         000111|      11001111| 33
|     0|         000000|      00000000| 34
|     1|         000001|      11010100| 35
|     4|         000100|      11101001| 36

----------------------RAM table|
|   RAM address |         data |
|              0|      00000000|
|              1|      11010100|
|              2|      00011100|
|              3|      10001100|
|              4|      11101001|
|              5|      00011011|
|              6|      01101011|
|              7|      11001111|
Request info: memory address - [000110] (6) | data - [10001111] (143)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record with the least number of appeals
Deleted el.[address; data; reference times]: 000111 | 11001111 | 1
Average access time: 14.3158; | Iteration num - 37 | Hit ratio - 0.324324
```



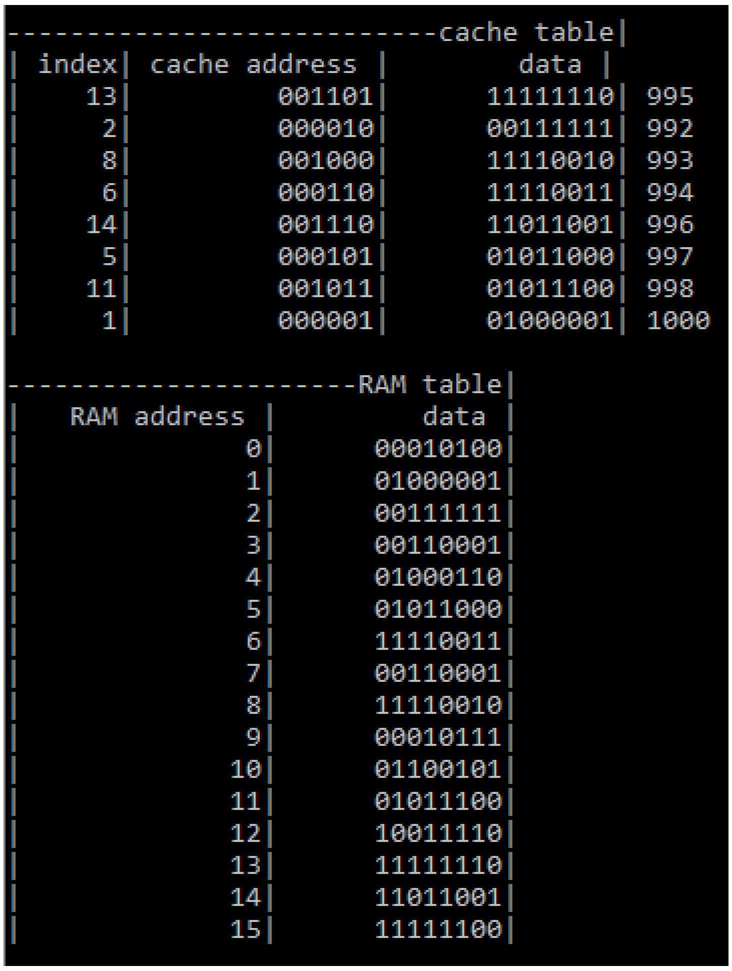
```
---------------------------cache table|
| index| cache address |        data |
|     0|         000000|      00000000| 34
|     1|         000001|      11010100| 35
|     4|         000100|      11101001| 36
|     6|         000110|      10001111| 37

----------------------RAM table|
|   RAM address |         data |
|              0|      00000000|
|              1|      11010100|
|              2|      00011100|
|              3|      10001100|
|              4|      11101001|
|              5|      00011011|
|              6|      10001111|
|              7|      11001111|
Request info: memory address - [000001] (1)
```



```
--------------------------cache table|
| index| cache address |        data |
|     7|         000111|      10010011| 0
|     2|         000010|      11010110| 1

---------------------RAM table|
|   RAM address |         data |
|              2|      11010110|
|              7|      10010011|
Request info: memory address - [000111] (7) | data - [01111001] (121)
***cache-hit***
--->>> write through: cache and RAM records rewriting
Average access time: 14; | Iteration num - 2 | Hit ratio - 0
Press c to continue, q to exit, i to display info and continue...^
t to show tables and coninue...

        Main info:
|- memory bit size: 6 -|
|- data bit size: 8 -|
|- RAM table size: 8 -|
|- cache table size: 4 -|
|- hit ratio - 0.333333 hit_n - 1 miss_n - 2 -|
|- T RAM - 20 T cache - 2 -|


--------------------------cache table|
| index| cache address |        data |
|     7|         000111|      01111001| 2
|     2|         000010|      11010110| 1

---------------------RAM table|
|   RAM address |         data |
|              2|      11010110|
|              7|      01111001|
```



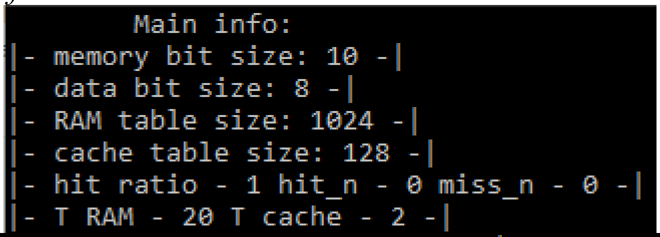
```
---------------------------cache table|
| index| cache address |        data |
|     7|         000111|      10010011| 0

----------------------RAM table|
|   RAM address |          data |
|              7|      10010011|
Request info: memory address - [000010] (2) | data - [11010110] (214)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 1 | Hit ratio - 0
Press c to continue, q to exit, i to display info and continue...^
t to show tables and coninue...

        Main info:
|- memory bit size: 6 -|
|- data bit size: 8 -|
|- RAM table size: 8 -|
|- cache table size: 4 -|
|- hit ratio - 0 hit_n - 0 miss_n - 2 -|
|- T RAM - 20 T cache - 2 -|


---------------------------cache table|
| index| cache address |        data |
|     7|         000111|      10010011| 0
|     2|         000010|      11010110| 1

----------------------RAM table|
|   RAM address |          data |
|              2|      11010110|
|              7|      10010011|
```



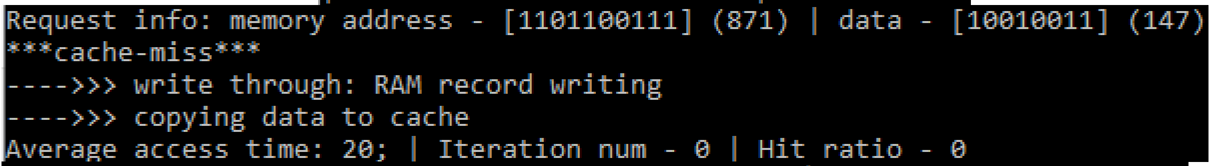
```
---------------------------cache table|
| index| cache address |        data |
|    13|         001101|      11111110| 995
|     2|         000010|      00111111| 992
|     8|         001000|      11110010| 993
|     6|         000110|      11110011| 994
|    14|         001110|      11011001| 996
|     5|         000101|      01011000| 997
|    11|         001011|      01011100| 998
|     1|         000001|      01000001| 1000

----------------------RAM table|
|   RAM address |          data |
|              0|      00010100|
|              1|      01000001|
|              2|      00111111|
|              3|      00110001|
|              4|      01000110|
|              5|      01011000|
|              6|      11110011|
|              7|      00110001|
|              8|      11110010|
|              9|      00010111|
|             10|      01100101|
|             11|      01011100|
|             12|      10011110|
|             13|      11111110|
|             14|      11011001|
|             15|      11111100|
```

Checking average access time and hit probability:

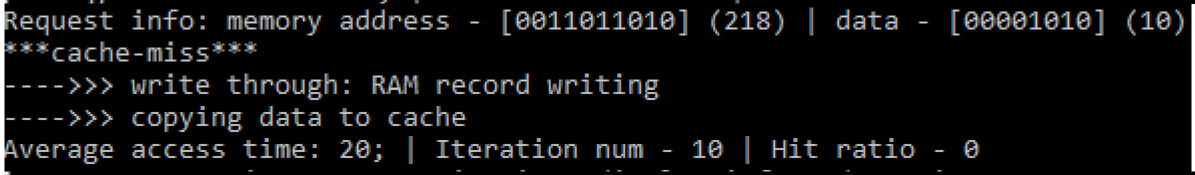
The ratio of the size of the RAM table to the cache is 1024/128:



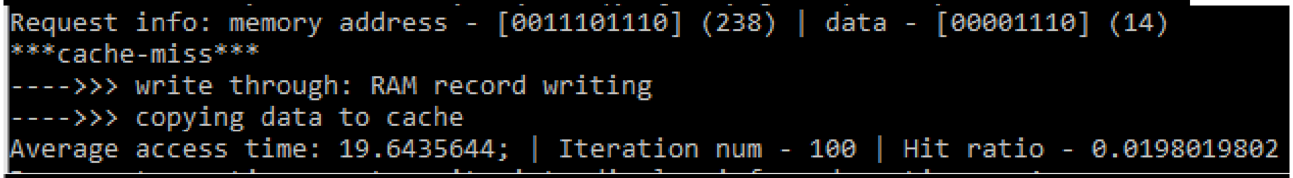
```
        Main info:
|- memory bit size: 10 -|
|- data bit size: 8 -|
|- RAM table size: 1024 -|
|- cache table size: 128 -|
|- hit ratio - 1 hit_n - 0 miss_n - 0 -|
|- T RAM - 20 T cache - 2 -|
```



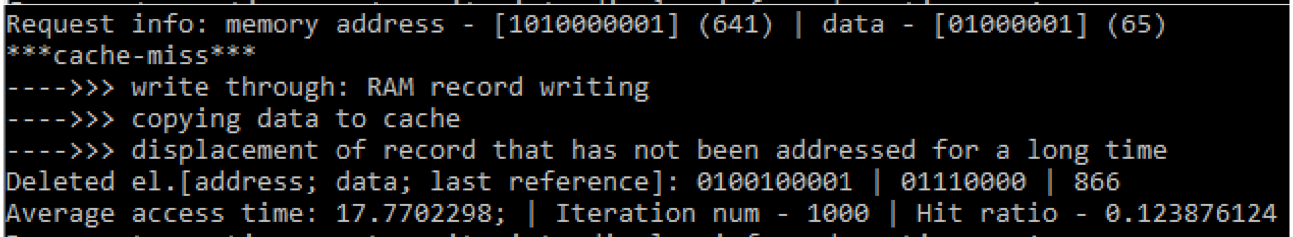
```
Request info: memory address - [1101100111] (871) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 0 | Hit ratio - 0
```



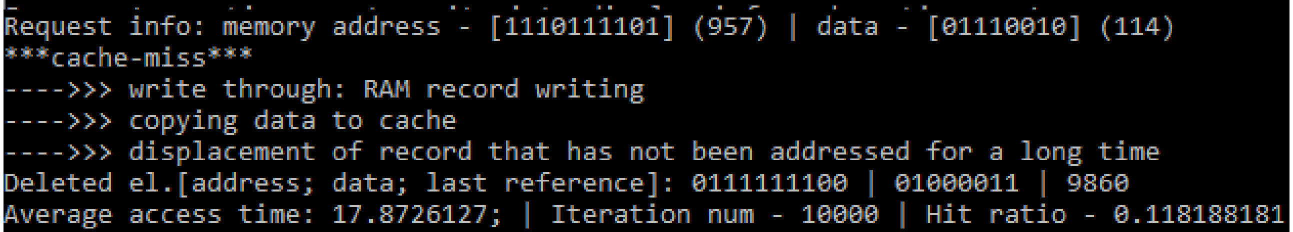
```
Request info: memory address - [0011011010] (218) | data - [00001010] (10)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 10 | Hit ratio - 0
```



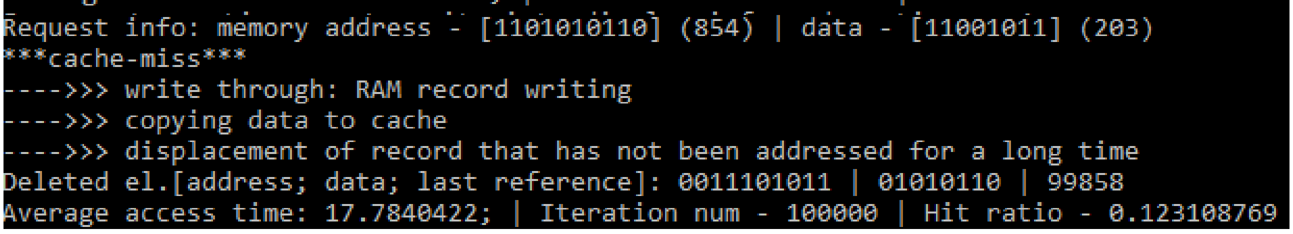
```
Request info: memory address - [0011101110] (238) | data - [00001110] (14)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 19.6435644; | Iteration num - 100 | Hit ratio - 0.0198019802
```



```
Request info: memory address - [1010000001] (641) | data - [01000001] (65)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 0100100001 | 01110000 | 866
Average access time: 17.7702298; | Iteration num - 1000 | Hit ratio - 0.123876124
```

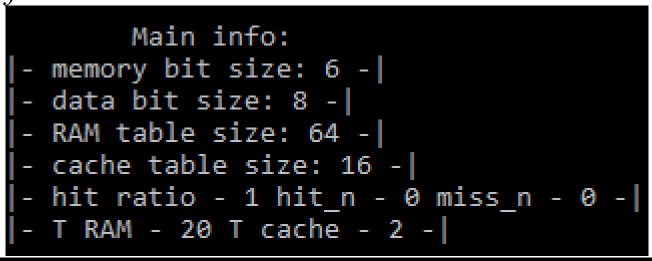


```
Request info: memory address - [1110111101] (957) | data - [01110010] (114)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 0111111100 | 01000011 | 9860
Average access time: 17.8726127; | Iteration num - 10000 | Hit ratio - 0.118188181
```

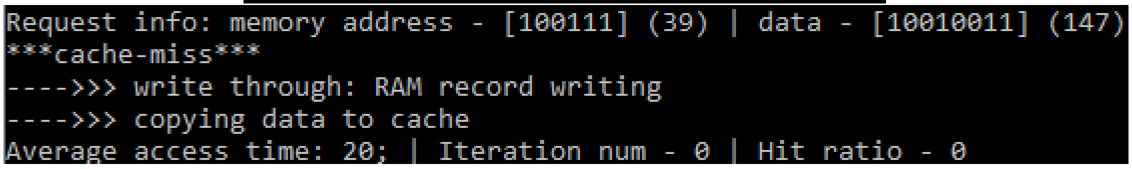


```
Request info: memory address - [1101010110] (854) | data - [11001011] (203)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 0011101011 | 01010110 | 99858
Average access time: 17.7840422; | Iteration num - 100000 | Hit ratio - 0.123108769
```

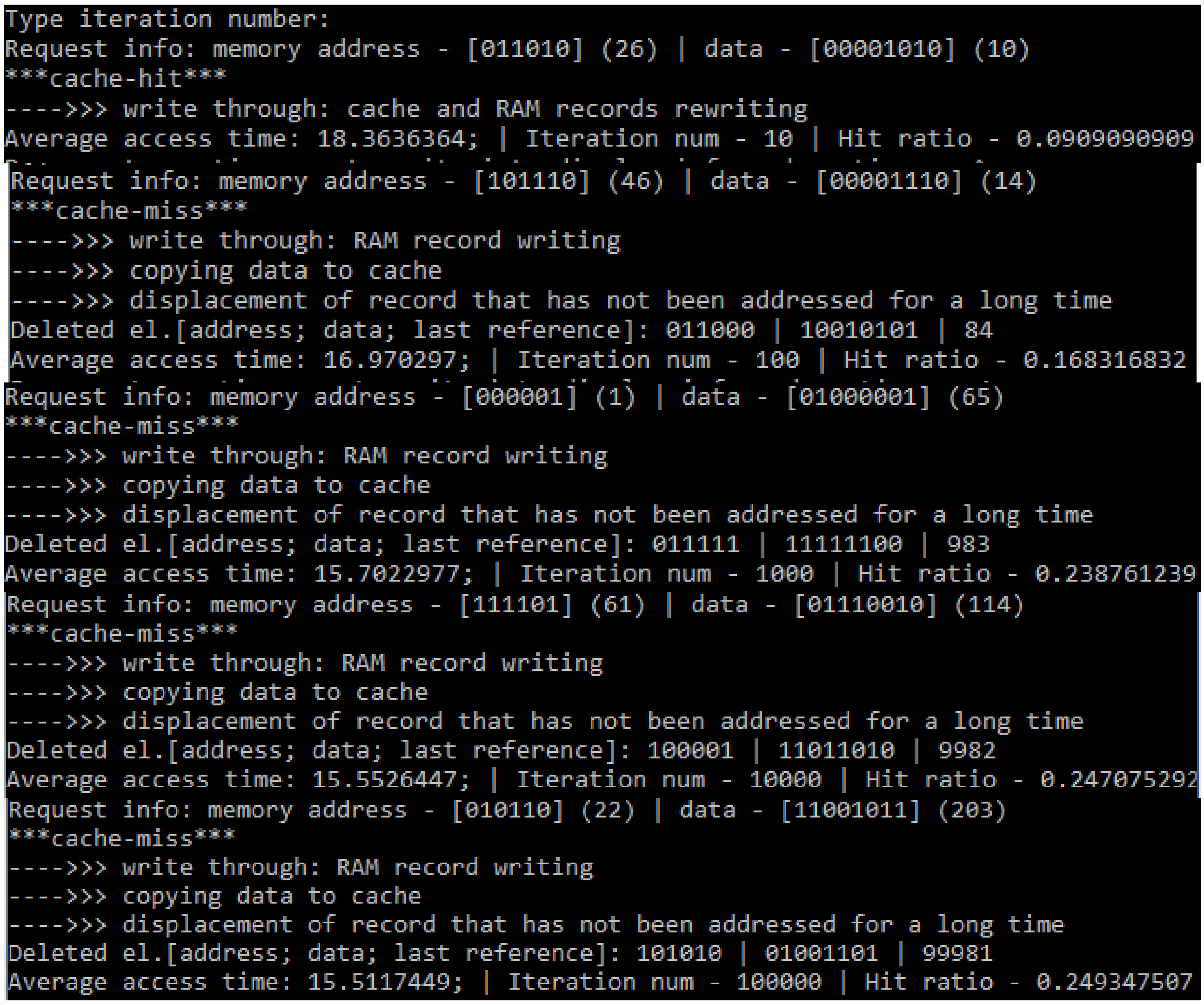
The ratio of the size of the RAM table to the cache is 64/16:



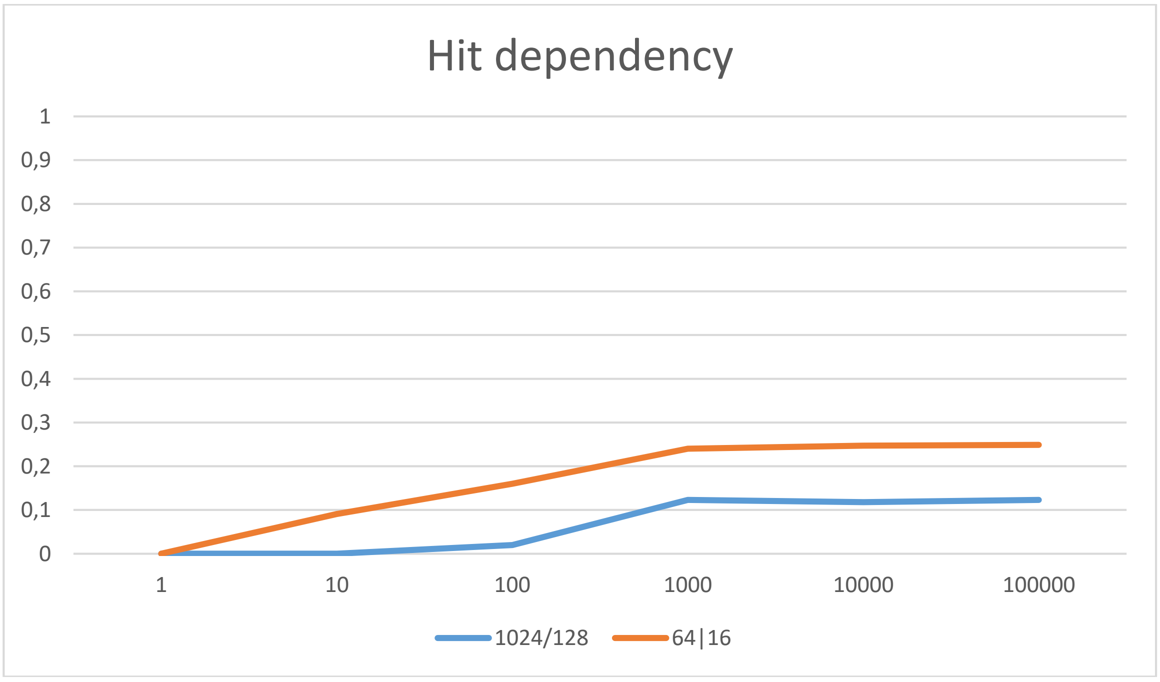
```
        Main info:
|- memory bit size: 6 -|
|- data bit size: 8 -|
|- RAM table size: 64 -|
|- cache table size: 16 -|
|- hit ratio - 1 hit_n - 0 miss_n - 0 -|
|- T RAM - 20 T cache - 2 -|
```



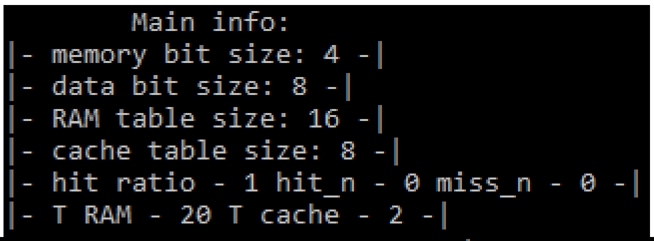
```
Request info: memory address - [100111] (39) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 0 | Hit ratio - 0
```



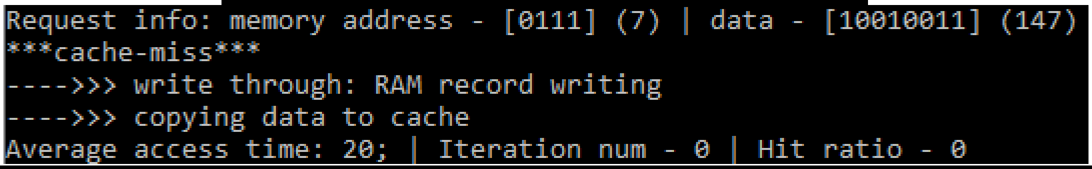
```
Type iteration number:
Request info: memory address - [011010] (26) | data - [00001010] (10)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 18.3636364; | Iteration num - 10 | Hit ratio - 0.0909090909
Request info: memory address - [101110] (46) | data - [00001110] (14)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 011000 | 10010101 | 84
Average access time: 16.970297; | Iteration num - 100 | Hit ratio - 0.168316832
Request info: memory address - [000001] (1) | data - [01000001] (65)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 011111 | 11111100 | 983
Average access time: 15.7022977; | Iteration num - 1000 | Hit ratio - 0.238761239
Request info: memory address - [111101] (61) | data - [01110010] (114)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 100001 | 11011010 | 9982
Average access time: 15.5526447; | Iteration num - 10000 | Hit ratio - 0.247075292
Request info: memory address - [010110] (22) | data - [11001011] (203)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 101010 | 01001101 | 99981
Average access time: 15.5117449; | Iteration num - 100000 | Hit ratio - 0.249347507
```



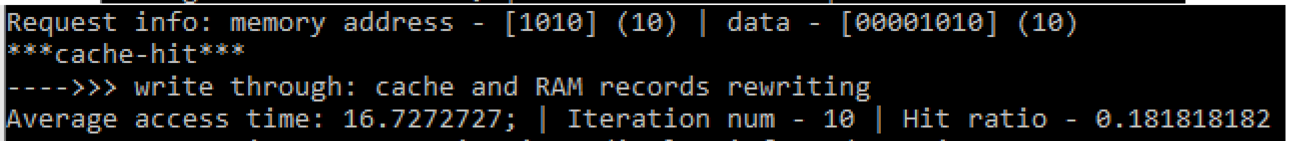
RAM to cache table size ratio - 16/8:



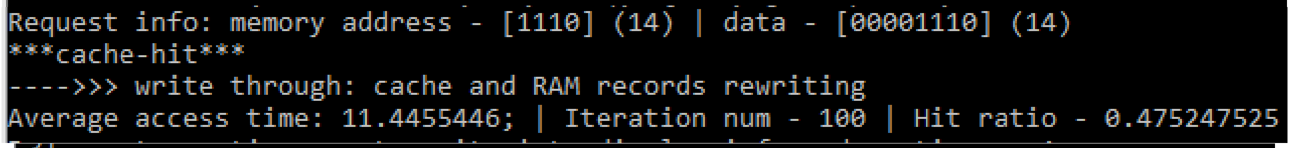
```
        Main info:
|- memory bit size: 4 -|
|- data bit size: 8 -|
|- RAM table size: 16 -|
|- cache table size: 8 -|
|- hit ratio - 1 hit_n - 0 miss_n - 0 -|
|- T RAM - 20 T cache - 2 -|
```



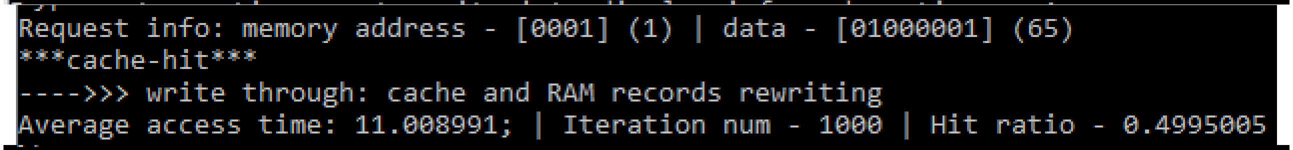
```
Request info: memory address - [0111] (7) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 0 | Hit ratio - 0
```



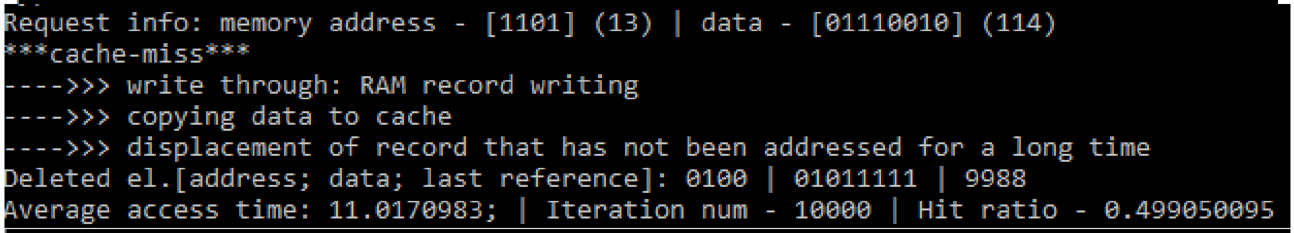
```
Request info: memory address - [1010] (10) | data - [00001010] (10)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 16.7272727; | Iteration num - 10 | Hit ratio - 0.181818182
```



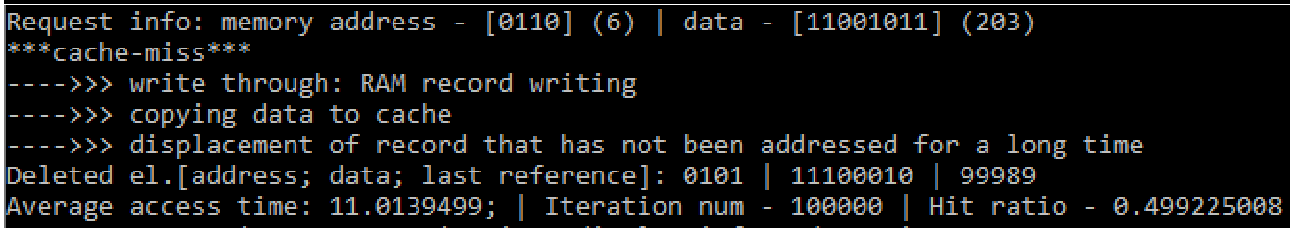
```
Request info: memory address - [1110] (14) | data - [00001110] (14)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.4455446; | Iteration num - 100 | Hit ratio - 0.475247525
```



```
Request info: memory address - [0001] (1) | data - [01000001] (65)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.008991; | Iteration num - 1000 | Hit ratio - 0.4995005
```

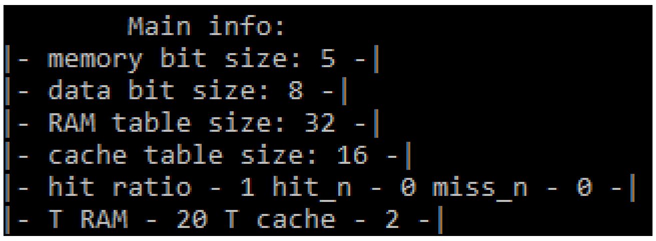


```
Request info: memory address - [1101] (13) | data - [01110010] (114)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 0100 | 01011111 | 9988
Average access time: 11.0170983; | Iteration num - 10000 | Hit ratio - 0.499050095
```

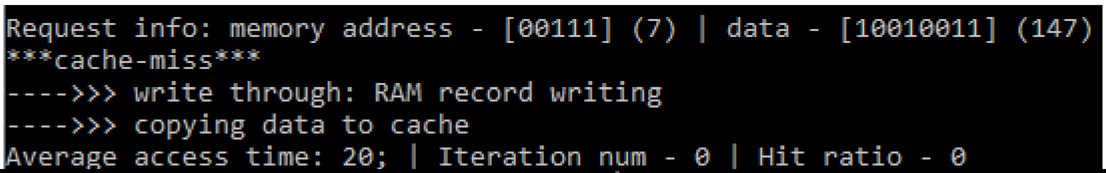


```
Request info: memory address - [0110] (6) | data - [11001011] (203)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 0101 | 11100010 | 99989
Average access time: 11.0139499; | Iteration num - 100000 | Hit ratio - 0.499225008
```

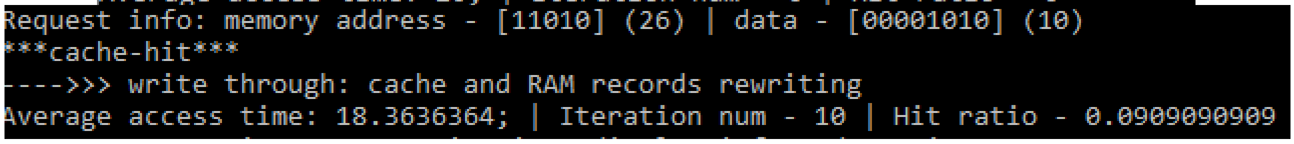
RAM to cache table size ratio – 32/16:



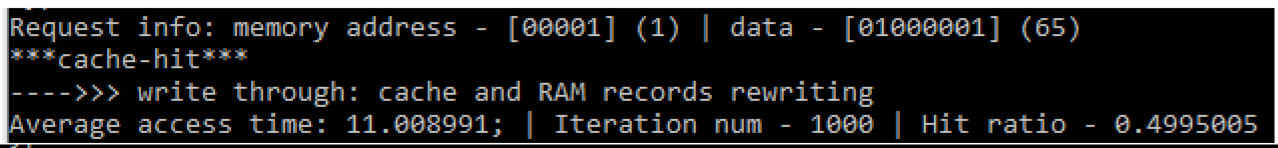
```
        Main info:
|- memory bit size: 5 -|
|- data bit size: 8 -|
|- RAM table size: 32 -|
|- cache table size: 16 -|
|- hit ratio - 1 hit_n - 0 miss_n - 0 -|
|- T RAM - 20 T cache - 2 -|
```



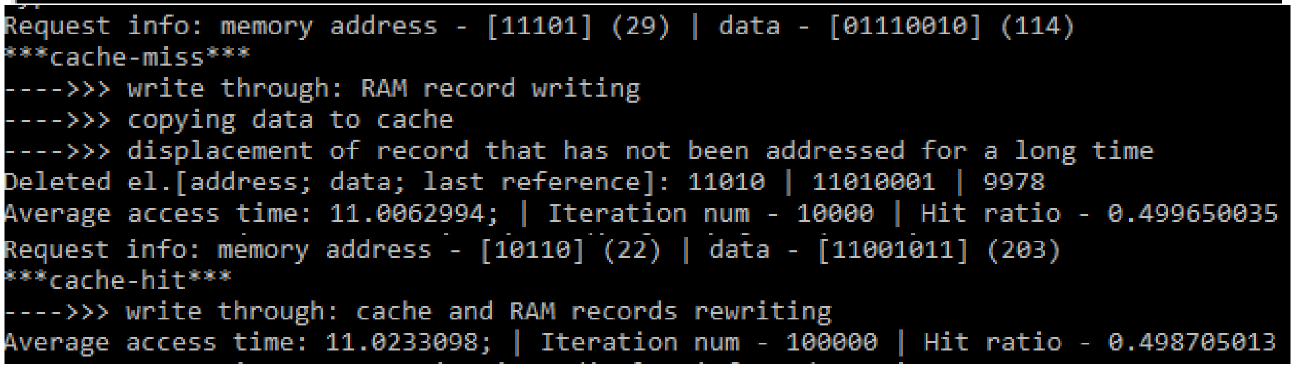
```
Request info: memory address - [00111] (7) | data - [10010011] (147)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
Average access time: 20; | Iteration num - 0 | Hit ratio - 0
```



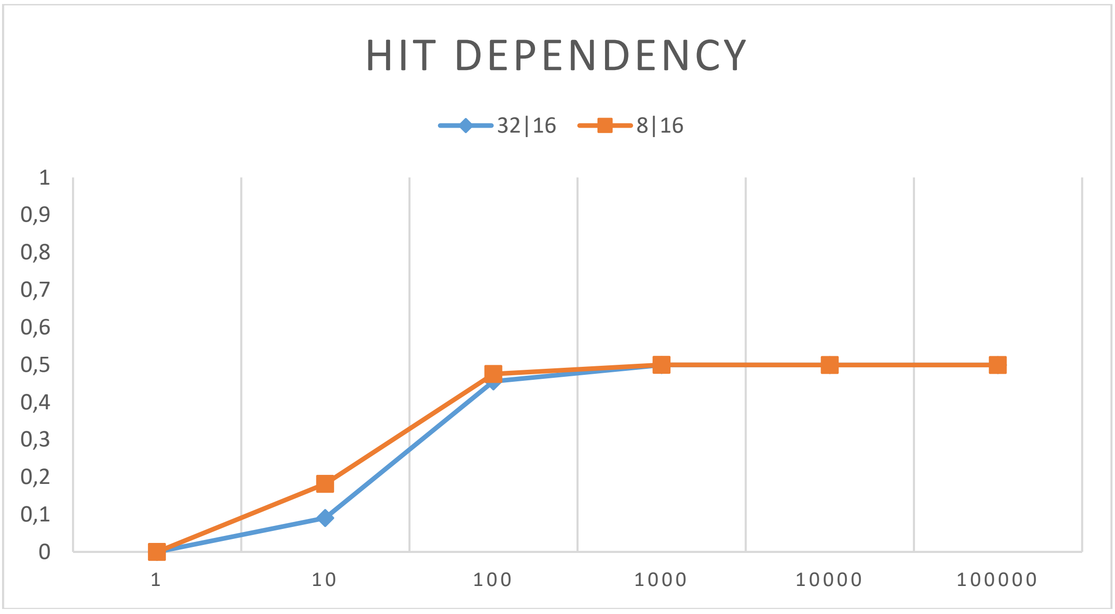
```
Request info: memory address - [11010] (26) | data - [00001010] (10)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 18.3636364; | Iteration num - 10 | Hit ratio - 0.0909090909
```



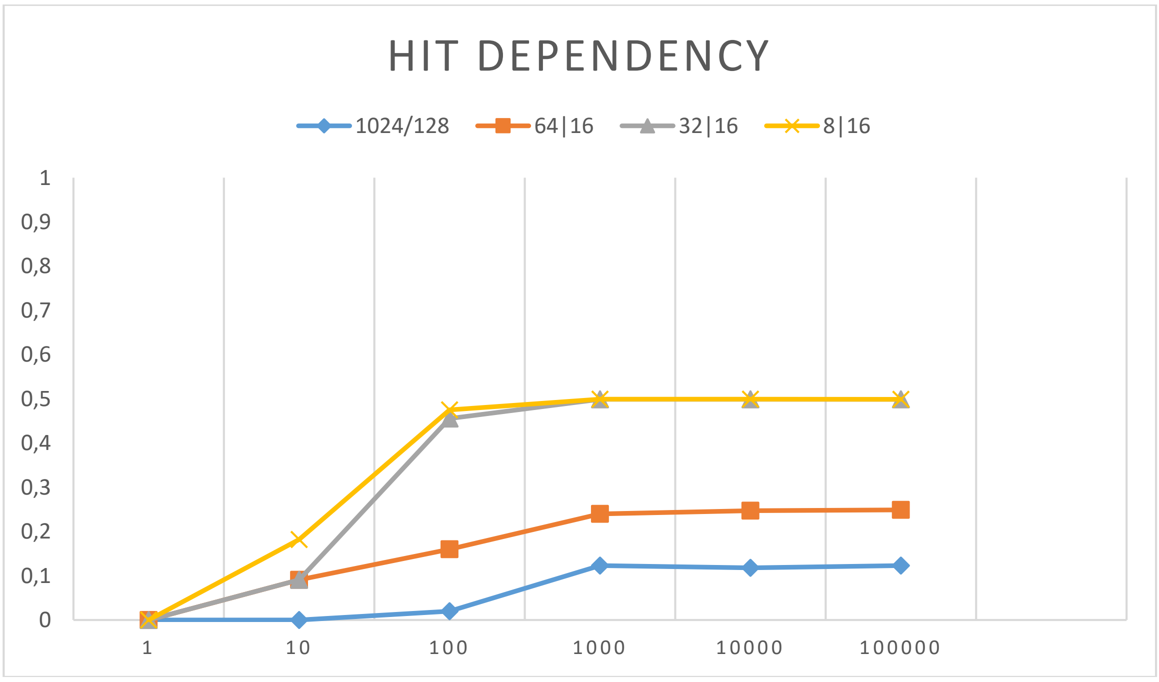
```
Request info: memory address - [00001] (1) | data - [01000001] (65)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.008991; | Iteration num - 1000 | Hit ratio - 0.4995005
```



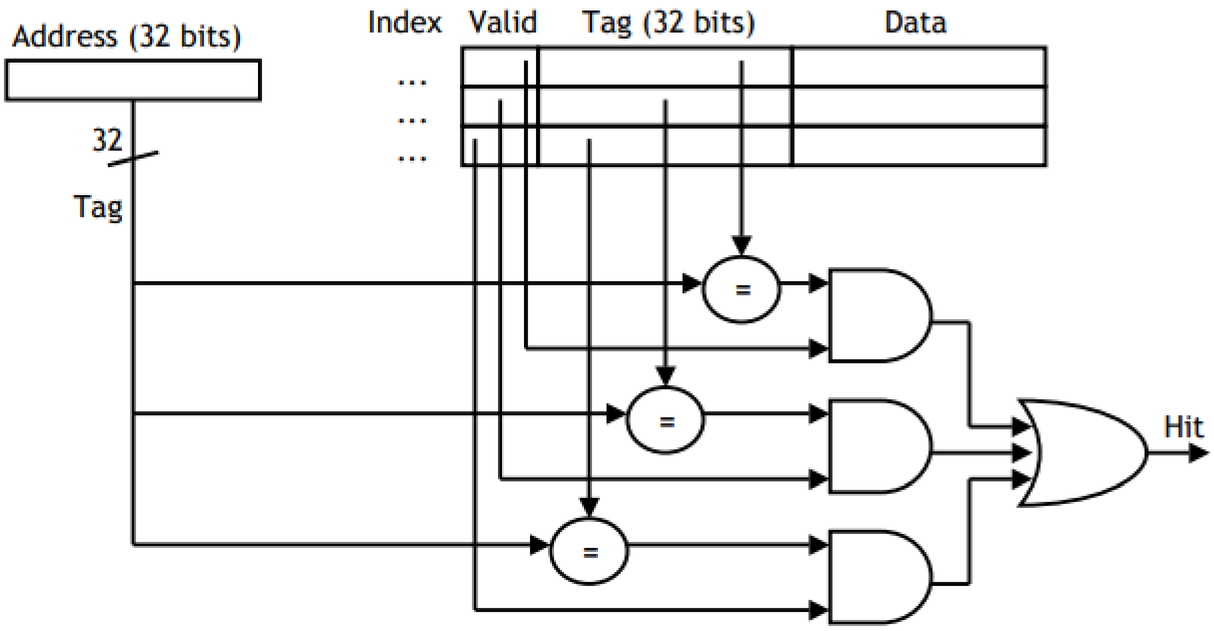
```
Request info: memory address - [11101] (29) | data - [01110010] (114)
***cache-miss***
---->>> write through: RAM record writing
---->>> copying data to cache
---->>> displacement of record that has not been addressed for a long time
Deleted el.[address; data; last reference]: 11010 | 11010001 | 9978
Average access time: 11.0062994; | Iteration num - 10000 | Hit ratio - 0.499650035
Request info: memory address - [10110] (22) | data - [11001011] (203)
***cache-hit***
---->>> write through: cache and RAM records rewriting
Average access time: 11.0233098; | Iteration num - 100000 | Hit ratio - 0.498705013
```



:



:



However, although the associative method is quite expensive to use, but the missing index field in the tag field, the entire address is used as a tag, which allows to increase the overall cache size. As well, this does not negate the fact that each cache block should be checked to find the required tag.

The problem of working with the cache is also how to minimize "cache misses", that is, attempts to read records that are not in the fast memory. Since too many misses reduce the performance of the cache. your cache is reduced. One way to mitigate this problem is to use larger memory sizes.

Another problem with caches is the risk of reading "stale" data when the data in the cache does not reflect the latest data in the underlying source. However, this risk is an acceptable trade-off for the performance of the application, and the application can update the corresponding cache entry if necessary.

During the testing of the program, the performance check of the program algorithm and the study of caching efficiency were carried out. For this, the average cache access time was calculated using the equation $t = t2p + t1(1-p)$, and the probabilities of hitting the cache were also determined. The first graph analyzed the distribution of probabilities depending on the number of passed iterations in the ratio of cache to RAM as 4:8, respectively. As you can see, up to the 240th iteration, the data distribution rose in efficiency, but then the graph stopped progressing and stopped at a result of ~50%, which is a rather bad result and is explained by the small size of the cache. A similar trend can also be seen in the second graph, which shows a gradual improvement in memory access time, with the rapid progress stopping after 240 iterations. In the 3rd graph, a trend of increasing efficiency with an increase in the number of iterations is noticeable, as in the first graph, the size of the cache exceeds the size of the RAM, and in the second graph, the size of the cache in the ratio of 128:1024 to the RAM shows an extremely low efficiency of interaction. The 4th graph tests the possibility of the dependence of the values of the probability of a cache hit on the scale of the size of the tables. From the conducted research, it was proved that the value of the ratio, which is 4/8 = 8/16 = 16/32 = 0.5 with a sufficient number of samples, does not affect the result of the probability of cash hits. That is why, in order to maintain a high level of performance and at the same time preserve the amount of cache and RAM in an efficient configuration, choose a value in accordance with cache memory to RAM in the range from 0 to 1, combining the ratio of table sizes in such a way as to achieve the most efficient interaction without losing the balance between cache and RAM.

The possibility of obtaining a high percentage of efficiency is determined by the presence of objective properties in the data, namely spatial and temporal locality. Spatial locality works in such a way that if an address has been accessed, it is highly likely that neighboring addresses will be accessed in the near future.

Temporal locality, in turn, says that after a call to a certain address, the next call will be in the near future with a high probability to the same address.