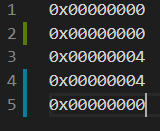
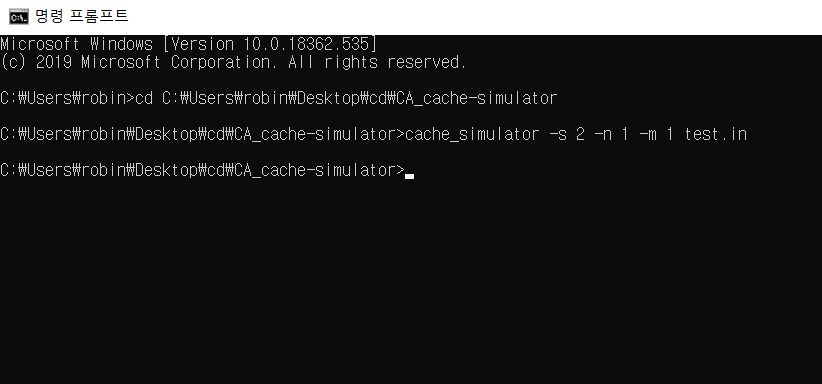
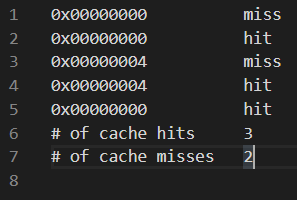


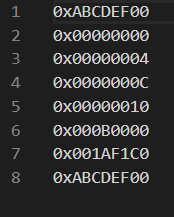
[**2019-2**](http://blog.naver.com/sangsang0852/70117334580) **Computer Architecture**

**Term Project Report**

[**< Simulator for N-way Set Associative Cache >**](http://blog.naver.com/sangsang0852/70117334580)

|  |  |
| --- | --- |
| **Course :** | **Computer Architecture** |
| **Instructor :** | **Hyosu Kim** |
| **Date :** | **2019. 12. 21** |
| **ID :** | **20184757** |
| **Name :** | **주영석** |

1. **Idea**  
    Implemented using C++. Make block as class, containing tag, pointer of block that points next block of set and order number in a set. When finding specific block in set, traverse from first block of set, using next block pointer. If the block is not the block you find, increment its order value as 1. When the block is the block you were finding, make the block come to front of the set by modifying the next block pointer value, which is idea of LRU algorithm.
2. **Implementation Detail** First, get command line argument and parse it as *s, n, m* value, and also parse input file name *(line 116~158)*. Then, open input file, and copy each line from input file to std::vector *(line 161, 48~64)*.  
    When input process is done, make cache using vector containing pointer of block which has size of s, std::vector<Block \*> cache(s); *(line 168)*. And, open output file to write, slicing input file name, changing filename extension from .in to .out *(line 169)*.   
    Now, read each element of vector containing memory access addresses. Convert the string into unsigned int which is 32-bit. Using this value, parse set index and tag with *s, m* input values and log functions from *#include <cmath>*. Then, use findBlock() function to check cache hit/miss. Write output file.
3. **Test**(1)

(2) 