Computer Architecture Project

-Implementation of Cache Simulator

20161166 양희웅

-summary:

There are so many ways to determine cache is hit or miss. One of them is to make ‘set associative cache’. We can make it by splitting cache blocks to several ways.

I made Cache Simulator which can expect hit or miss rate of addresses set.

Language: C

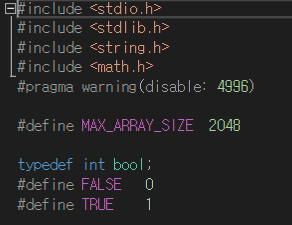
Compile Environment: Linux

Name of binary file: Cache\_Simulator

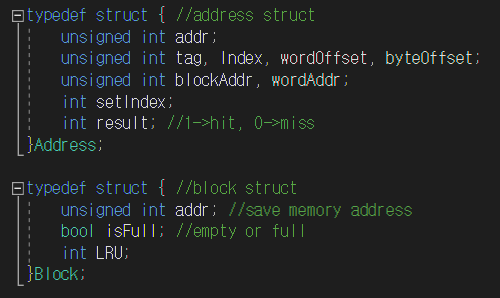
Input file: test.txt

How to use: In linux console, regardless of m, n, s order, ./Cache\_Simulator -m <M> -n <N> -s <S> test.txt

-details:



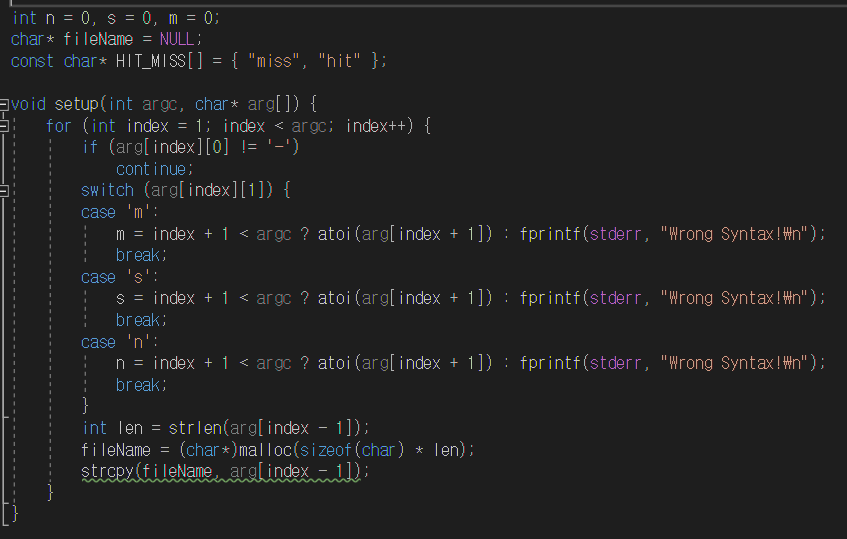
I set max address number as 2048. I tried to allocate address number. But is was so difficult that I took several times but failed. I make another type: bool for state ‘isFull’ and ‘isHit’.



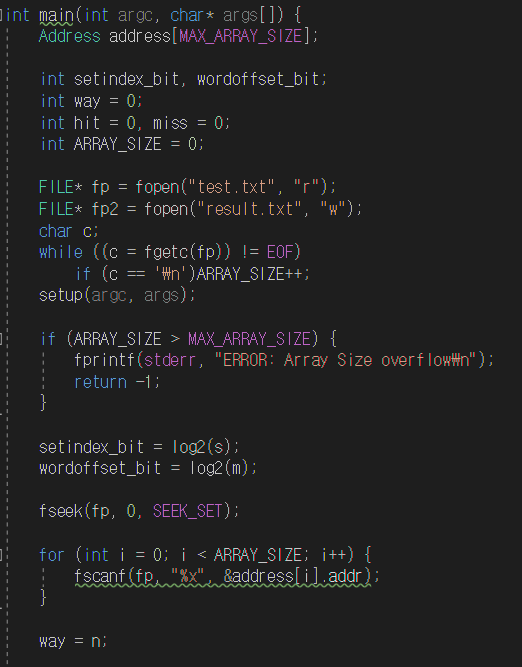
I made a structs for address and block.

Addr. Struct contains information about address: addr, tag index, word offset, byte offset, block address, word address, set index, result.

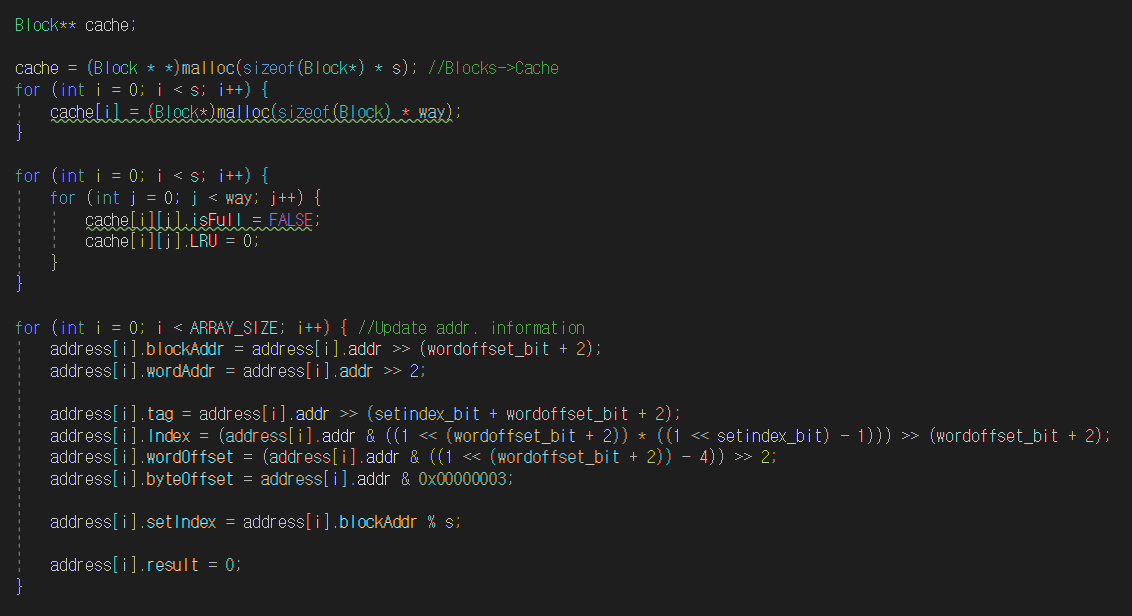
Block struct contains information about block: addr, whether this block is full or not, LRU(Least-Recently Used).



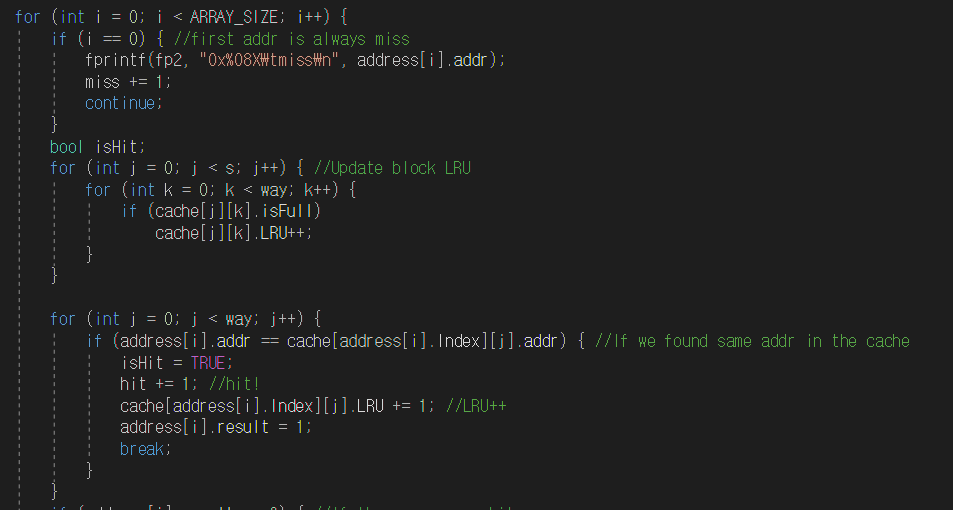
This part is for getting information about cache. At command, users can enter information about cache such as the number of sets in a cache, the number of sets in a set, the number of words in a block. And I distinguished it using ‘-s’, -m’, ‘-s’ tag for each information, also input file name.



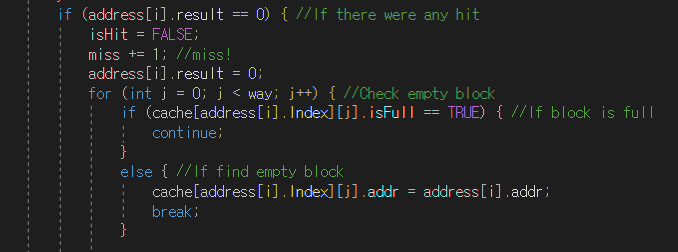
This is initializing part. First, I make file stream for input file and output file. I get number of addresses using fgetc. I read file names until meet EOF. We type number of set and number of words by decimal number. We already typed number of sets and number of blocks in a set. So, number of sets becomes number of row and number of blocks in a set becomes number of columns. And also, I use log function to decide it’s offset bit.

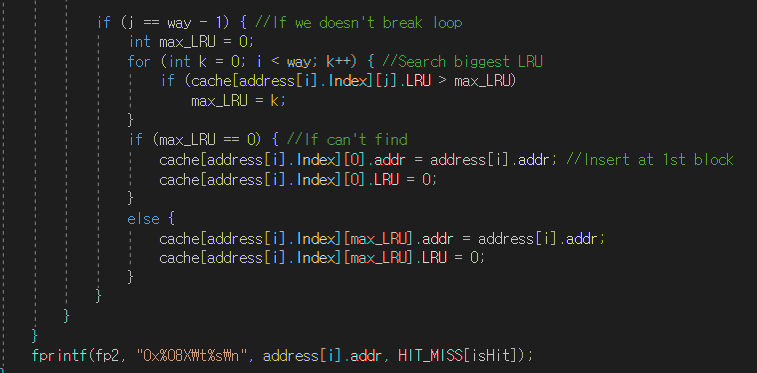


This is initializing part 2. I allocated 2-dimensional block array using malloc. And initialized it. At first, blocks are empty. So isFull state is False and LRU is also 0. Address information are initialized by shift operation. I refer memory address structure in class file.

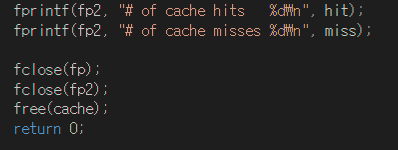


This is determining part. Maybe it is the most important part of the cache simulator. First address is must miss and I made separate case for it. Next, I update LRU of blocks which is full. Next part is to determine ‘Hit’ case. At first, it searches address’s set index row. If it finds same tag at there, it means hit. And change isHit state as TRUE, change result state, add LRU number.



In this part, we determine miss case and do the right actions. First, change isHit state to FALSE, add miss count, change result state as 0 one more time. Then we search empty block in that set. If we find empty block, we fill it with current address. 

At this part, we do actions when we can’t find empty space. First, we find the biggest LRU block. If we find that, insert current address in there. But if we doesn’t, when every block’s LRU is same, I insert current address at first block. Finally, states is written by fprintf function.



At last, hit and miss counts are written at file by fprintf function. And close file stream, release allocated memory space.