One Possible Design for Programming Assignment 2 Cache Simulation

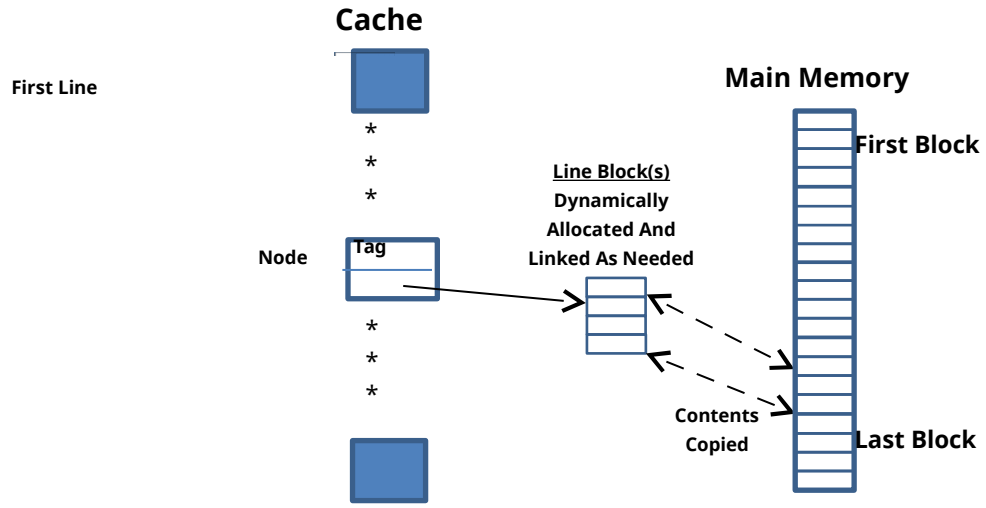
The following drawing is a suggested organization for this assignment. Of course you may implement it as you wish, provided somewhere in your program you use a *structure*. Note the following:

1. The cache is an array of nodes where a node is defined as:

```
struct node {
  int   tag;          //determined by location of main memory block
  int   *lineBlock;   //points to a dynamically allocated line-block
}
```

   Additional information on using the struct can be found in the C folder I posted on Moodle or in the C references I gave the class earlier.

2. Whenever a block of main memory is referenced, a block is allocated and "attached" to the cache. Then the content of the referenced memory block is copied into that block.
3. Whenever a value is written into an allocated block in cache, that same value is also written back into main memory (Write Through). In this way, if another main memory block gets assigned into a cache line already occupied by another block, the new block's values just overwrite the existing block's values. Of course the tag must be set, too.



4. In the above drawing, the cache array and main memory array are dynamically allocated immediately after the user enters the required information to do so (i.e. memory size, cache size and block size). All the nodes in the cache are then initialized with tags which are invalid (e.g. -1) and their line-block pointers are set to NULL. Main memory's content is initialized as described in the assignment's specification.
5. Every time an empty node (i.e. line) in the cache is referenced, a new "block" is dynamically allocated and linked to that node using the *lineBlock* pointer. The appropriate *tag* value is inserted in the respective node and the contents of the main memory block represented by the new linked-block is copied into the block. At this point, the block in main memory and the linked line-block have the same content.

6. When the user quits, all dynamically allocated arrays and blocks must be freed up.

Miscellaneous Pieces of Code

Define node-type

```
struct node{
  int tag;
  int *block;       //pointer to dynamic array
} *cache = NULL;

typedef struct node n
```

Allocate and initialize main memory

```
  /*allocate space for MM and initialize*/
  mm = (int*)malloc(mmSize * sizeof(int));
  for(i = 0; i < mmSize; i++){
    mm[i] = mmSize - 1;
```

Allocate and initialize cache memory

```
/*allocate space for cache and initialize*/
cache = (n*)malloc(numLines * sizeof(n));
for(i = 0; i < cacheSize; i++){
  cache[i].block = NULL;
}
```