Given a cache hierarchy with 8 words with a cache access time of 5 ns and a memory access time of 50ns, and the following set of memory accesses representing the tag and index: 22, 28, 31, 22, 16, 13, 31, 8, 5, 13, 8, and 5.

a) If the cache is a direct-mapped cache, show the sequence for accessing the cache and calculate the total access time. Draw the final cache table

b) If the cache is a 2-way set associative cache, show the sequence for accessing the cache and calculate the total access time.

c) Determine which cache has the better performance and by how much.

Average Memory Access Time

Given a Level-1 cache with a hit rate of 95%, a cache access time of 5 ns, and a memory access time of 100ns, find the average memory access time

The processor in the Example is given a L2 cache with a hit rate of 85%, and a cache access time of 7ns. Find the average memory access time

The processor in the Example is given a L3 cache with a hit rate of 80%, and a cache access time of 10ns. Find the average memory access time

Given the following code segments, annotate the code to describe what changes can be made to the code to improve cache memory access time (both data and instruction), memory consumption, and branching, as well as why the change results in the improvement. (You do not need to re-write code for full credit).

```
#include <time.h>
#include <stdio.h>
#include <stdlib.h>

typedef struct example{
    int int1;
    double double1;
    float float1;
    double double2;
    int int2;
}example;

int func( int count, example* the_arr, long unsigned int index ){
    return count += the_arr[index].int1;
}

int main( const int argc, const char* argv[] ){

    int array_size = atoi( argv[1] ), sum = 0;

    example* the_list = (example *)calloc(array_size,sizeof(example));

    int i;
    for(i = 0; i < array_size; ++i){
        the_list[i].int1 = i;
        the_list[i].double1 = (double)i;
        the_list[i].float1 = (float)i;
        the_list[i].double2 = (double)(i+1);
        the_list[i].int2 = i+1;
    }

    int test;
    for(test = 0; test < 4; ++test){

        long unsigned int index;
        for( index = 0; index < array_size; ++index ){

            the_list[index].int1 += func( test, the_list, index );
            sum += the_list[index].int2;
        }
    }

    free( the_list );

    return EXIT_SUCCESS;
}
```