C/C++ compatible Skeleton:

```c
#include <stdio.h>
#include <stdbool.h>

// Constants for the cache configuration
#define NUM_SETS /* Configure number of sets */
#define ASSOC /* Configure cache associativity */
#define BLOCK_SIZE 64 // 64 bytes as block size

// Define the arrays for the cache simulation
long long int tag_array[NUM_SETS][ASSOC];
long long int lru_position[NUM_SETS][ASSOC];
bool dirty[NUM_SETS][ASSOC];

// Variables to maintain the simulation statistics
int Hit = 0;
int Miss = 0;

// Forward declarations
void Update_lru(long long int add);
void Update_fifo(long long int add);

// Function to simulate cache access
void Simulate_access(char op, long long int add) {
    int set = (add / BLOCK_SIZE) % NUM_SETS;
    long long int tag = add / BLOCK_SIZE;

    for(int i = 0; i < ASSOC; i++) {
        if(tag == tag_array[set][i]) {
            // Cache hit scenario
            Hit++;
            // Choose policy (LRU or FIFO) based on the configuration
            if(/* LRU policy is chosen */) {
                Update_lru(add);
            } else {
                Update_fifo(add);
            }
        } else {
            // Cache miss scenario
            Miss++;
            // Handle the miss scenario here
        }
    }
}

// Update functions for different policies
void Update_lru(long long int add) {
    // Logic for updating LRU policy
}

void Update_fifo(long long int add) {
    // Logic for updating FIFO policy
}

int main() {
    char op;
    long long int add;
    FILE *file = fopen(/* Path to the trace file */, "r");

    // Check if the file opened successfully
    if(!file) {
        printf("Error: Could not open the trace file.\n");
        return 1;
    }

    // Read until end of file
    while(!feof(file)) {
        // Read operation and address
        fscanf(file, " %c %llx", &op, &add);

        // Begin the simulation for each address read
        Simulate_access(op, add);
    }

    // Print out the statistics
    printf("Hits: %d\n", Hit);
    printf("Misses: %d\n", Miss);

    return 0;
}
```