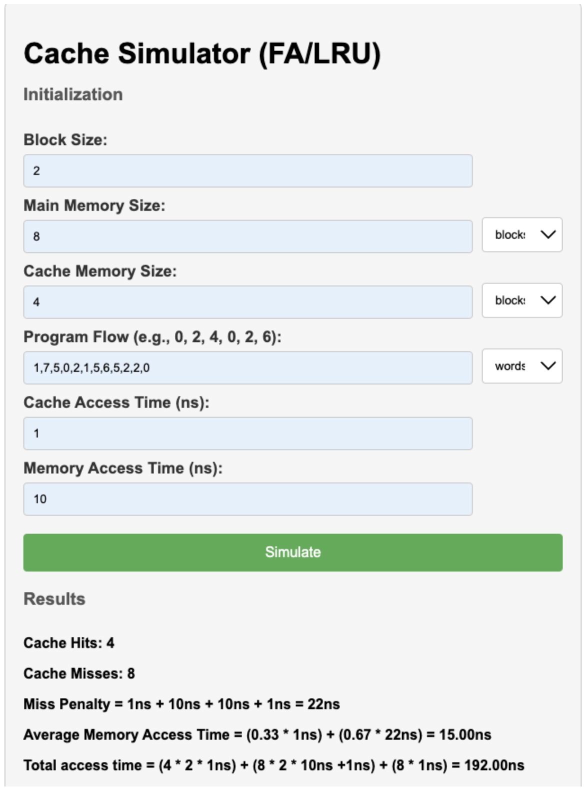
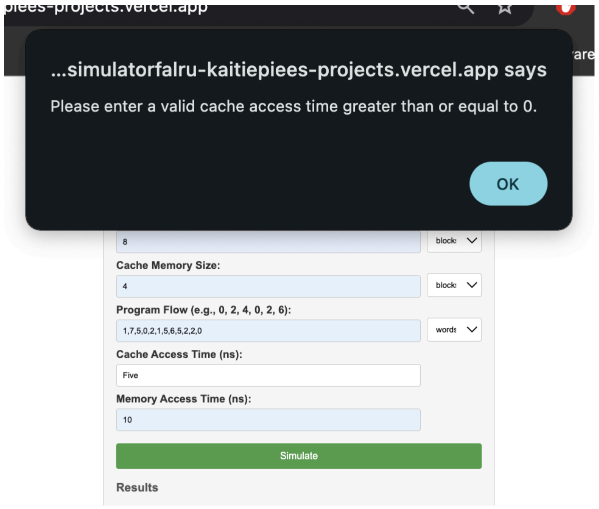
Group 4 - Cache simulator (Full associative / LRU)

Program Output Debugging

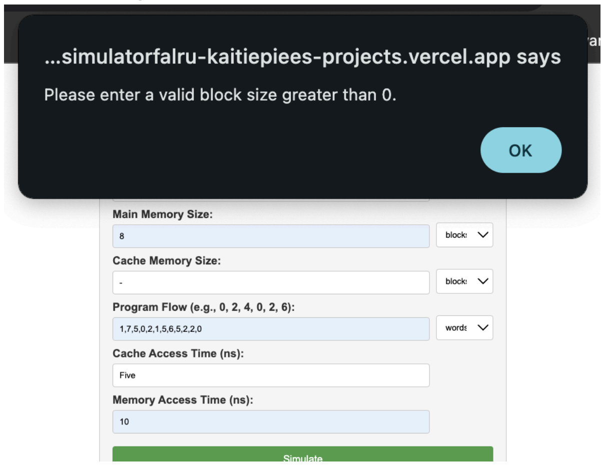
Test Case 1 (Normal Case)

Input: Block Size: 2 Main Memory Size: 8 Cache Memory Size: 4 Program Flow: 1,7,5,0,2,1,5,6,5,2,2,0 Cache Access Time(ns): 1 Memory Acces Time(ns): 10	Actual Output:  **Cache Simulator (FA/LRU)** Initialization Block Size: 2 Main Memory Size: 8 blocks Cache Memory Size: 4 blocks Program Flow (e.g., 0, 2, 4, 0, 2, 6): 1,7,5,0,2,1,5,6,5,2,2,0 words Cache Access Time (ns): 1 Memory Access Time (ns): 10 Simulate Results Cache Hits: 4 Cache Misses: 8 Miss Penalty = 1ns + 10ns + 10ns + 1ns = 22ns Average Memory Access Time = (0.33 * 1ns) + (0.67 * 22ns) = 15.00ns Total access time = (4 * 2 * 1ns) + (8 * 2 * 10ns +1ns) + (8 * 1ns) = 192.00ns
Expected Output: Cache Hits: 4 Cache Misses: 8 Miss Penalty = 1ns + 10ns + 10ns + 1ns = 22ns Average Memory Access Time = (0.33 * 1ns) + (0.67 * 22ns) = 15.00ns Total access time = (4 * 2 * 1ns) + (8 * 2 * 10ns +1ns) + (8 * 1ns) = 192.00ns	

Test Case 2 (Different Input)

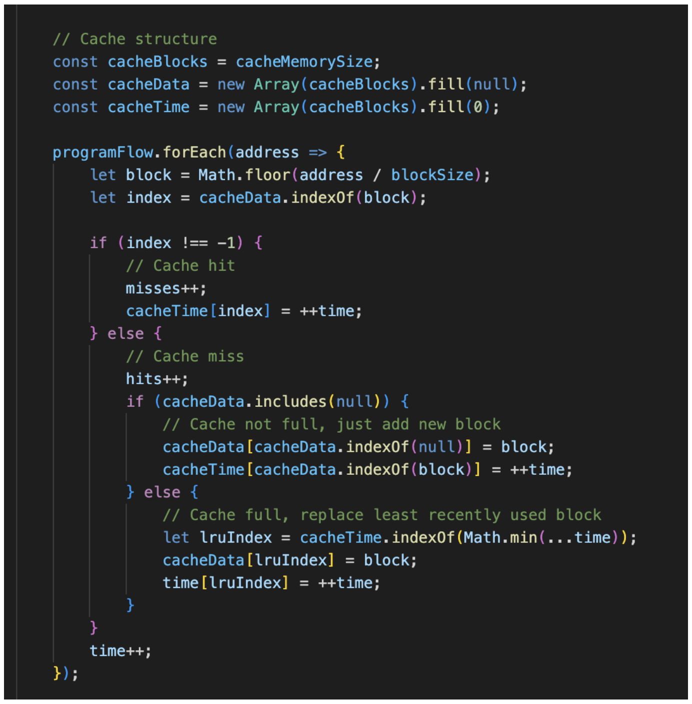
Input: Block Size: 2 Main Memory Size: 8 Cache Memory Size: 4 Program Flow: 1,7,5,0,2,1,5,6,5,2,2,0 Cache Access Time(ns): five Memory Acces Time(ns): 10	Actual Result:  piees-projects.vercel.app ...simulatorfalru-kaitiepiees-projects.vercel.app says Please enter a valid cache access time greater than or equal to 0. OK 8 blocks Cache Memory Size: 4 blocks Program Flow (e.g., 0, 2, 4, 0, 2, 6): 1,7,5,0,2,1,5,6,5,2,2,0 words Cache Access Time (ns): Five Memory Access Time (ns): 10 Simulate Results
Expected Result: Please enter a valid cache access time greater than or equal to 0.	

Test Case 3 (Incomplete Input)

Input: Block Size: - Main Memory Size: 16 Cache Memory Size: - Program Flow: 1,2,3,4,5,6,7 Cache Access Time(ns): 1 Memory Acces Time(ns): 10	Actual Output: ...simulatorfalru-kaitiepiees-projects.vercel.app says Please enter a valid block size greater than 0. Main Memory Size: 8 blocks Cache Memory Size: - blocks Program Flow (e.g., 0, 2, 4, 0, 2, 6): 1,7,5,0,2,1,5,6,5,2,2,0 words Cache Access Time (ns): Five Memory Access Time (ns): 10
Result: Please enter a valid block size greater than 0. Please enter a valid cache memory size greater than 0.	

Program Analysis:

The website is a Cache simulator designed for Block-set-associative in Least Recently Used. The user will input the size of the block, main memory, cache memory, and the program flow followed by the access time of cache and memory. The program will calculate the cache hits and misses, its miss penalty, average and total memory access time, and a cache snapshot from the given input. The user also has an option to download the text file of the recently calculated values.

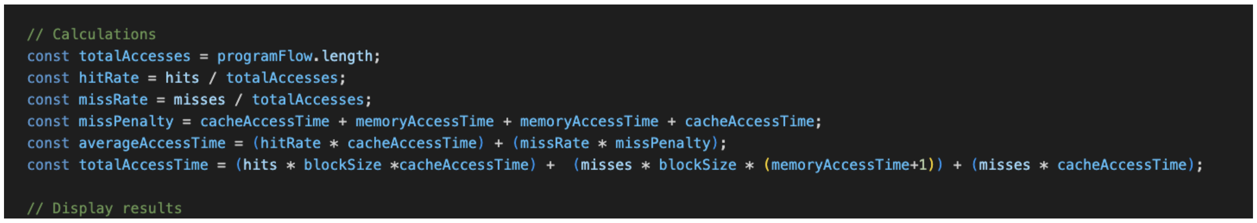


```
// Cache structure
const cacheBlocks = cacheMemorySize;
const cacheData = new Array(cacheBlocks).fill(null);
const cacheTime = new Array(cacheBlocks).fill(0);

programFlow.forEach(address => {
    let block = Math.floor(address / blockSize);
    let index = cacheData.indexOf(block);

    if (index !== -1) {
        // Cache hit
        misses++;
        cacheTime[index] = ++time;
    } else {
        // Cache miss
        hits++;
        if (cacheData.includes(null)) {
            // Cache not full, just add new block
            cacheData[cacheData.indexOf(null)] = block;
            cacheTime[cacheData.indexOf(block)] = ++time;
        } else {
            // Cache full, replace least recently used block
            let lruIndex = cacheTime.indexOf(Math.min(...time));
            cacheData[lruIndex] = block;
            time[lruIndex] = ++time;
        }
    }
    time++;
});
```

This program was designed using javascript, and CSS to style and layout the web page. The webpage is deployed using Vercel through this link: https://csarch2spcachesimulatorfalru-kaitiepiees-projects.vercel.app/ and is also runnable locally using CMD.



```
// Calculations
const totalAccesses = programFlow.length;
const hitRate = hits / totalAccesses;
const missRate = misses / totalAccesses;
const missPenalty = cacheAccessTime + memoryAccessTime + memoryAccessTime + cacheAccessTime;
const averageAccessTime = (hitRate * cacheAccessTime) + (missRate * missPenalty);
const totalAccessTime = (hits * blockSize *cacheAccessTime) +  (misses * blockSize * (memoryAccessTime+1)) + (misses * cacheAccessTime);

// Display results
```

One limitation of this cache simulator is that all inputs are required to run the simulation, making it less flexible for users who might want to test certain conditions without providing every single detail. Additionally, the input validation could be improved to ensure better error checking and prevent invalid from causing issues during the simulation.

Through this project, we gained a deep understanding of FA cache mapping and the LRU replacement algorithm by testing various memory access patterns. We observed how FA mapping allows any memory block to be loaded into any cache line, and how LRU effectively manages cache replacements by evicting the least recently accessed blocks. We also realized the differences between LRU and the Most Recently Used (MRU) algorithm. While LRU generally provides better performance by keeping frequently accessed data longer, MRU can sometimes be beneficial in specific scenarios. This hands-on experience highlighted the differences between these algorithms and deepened our understanding of cache management in computer architecture.