Direct Cache Mapping Simulator

PADILLA, Kai
De La Salle University
1st line of address
2nd line of address
Telephone number, incl. country code
email@dlsu.edu.ph

MARIÑAS, Carl Mervyn G.
De La Salle University
1st line of address
2nd line of address
Telephone number, incl. country code
carl_marinas@dlsu.edu.ph

EVANGELISTA, Reginald Andre I.
De La Salle University
1st line of address
2nd line of address
Telephone number, incl. country code
andre_evangelista@dlsu.edu.ph

1. INTRODUCTION

Computers today follow the Von Neumann Architecture which comprises the CPU, the memory, input-output devices, and the system bus. Memory stores the data and CPU communicates with the Memory to access, store, and fetch data. By separating the cpu and memory we can lessen the load on the cpu and increase operation speeds. Under memory there are things called cache memory and main memory. According to the Merriam-Webster dictionary cache is defined as "a computer memory with very short access time used for storage of frequently or recently used instructions or data". Cache or cache memory is a memory system that is responsible for temporarily storing data for quicker access by the CPU. Main memory and cache operate similarly but cache operates faster than main memory which also makes it more costly.

Cache mapping is a technique used to determine where data from the main memory will be stored in the cache memory. This is crucial because it bridges the speed gap between the faster CPU and the slower main memory, ensuring that both data retrieval and overall system performance are efficient. Cache mapping is essential for speed as it allows the CPU to access frequently requested data quickly. It also optimizes the use of cache memory, prioritizing efficiency and reducing the time the CPU spends waiting for data from the main memory. Lastly, it also helps in organizing data in the cache, making it easier to locate and retrieve. There are three primary types of cache mapping techniques, direct, full associative, and block set associative mapping. Direct cache mapping works where the main memory, divided into equal-sized blocks, is mapped onto the cache blocks in a modulo fashion. In this case, the modulo operation is used to determine the cache line where a particular block of memory will be placed; it ensures that the index value wraps around within the range of available cache lines. This way, every memory block is mapped to a valid cache line index.

2. Making of the program

```JavaScript
let mat = 10;
let cat = 1;
```

In this Direct Cache mapping simulator the Memory Access Time and the Cache Access Time is to a fixed value.

'cat' (Memory Acess Time) = 10 nanoseconds

'cat' (Cache Acess Time) = 1 nanosecod

```JavaScript
let hits = 0;
let misses = 0;
let cache = new
Array(block_size).fill(null);
for (let i = 0; i <
sequence_array.length; i++) {
let block = sequence_array[i];
let cache_block = block % cache_size;
if (cache[cache_block] === block) {
hits++;
} else {
misses++;
cache[cache_block] = block;
}
}
```

The block number from the memory access sequence is mapped to a specific cache line using the modulus operation : 'cache_block

= block % cache_size’. If the block is found the cache (‘cache[cache_block] === block’) it is considered a hit and it increments the hits variable. If the block is not found it is considered a miss then the block value will be stored on that specific cache block and incrementing the misses variable.

```
JavaScript
let miss_penalty = (2 * cat) +
(block_size * mat);

let amat = (hits /
sequence_array.length) * cat +
(misses / sequence_array.length) *
miss_penalty;

let tmat = (hits * block_size * cat)
+ (misses * block_size * (mat + cat))
+ (misses * cat);
```

In this part of the code is where the calculations of the Miss penalty, Average memory access time, and Total memory access time.

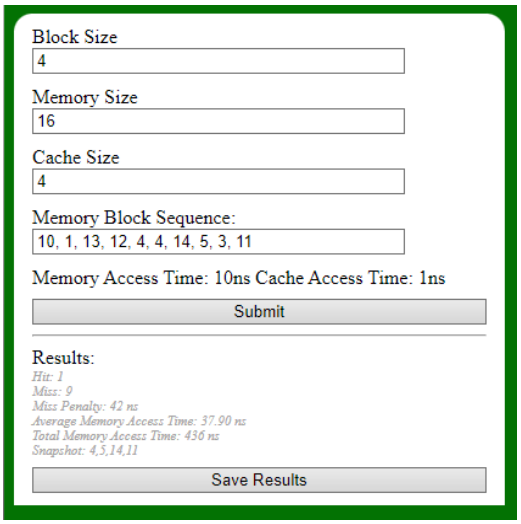
Miss_penalty = 2 * Cache access time + block_size * Memory Access time

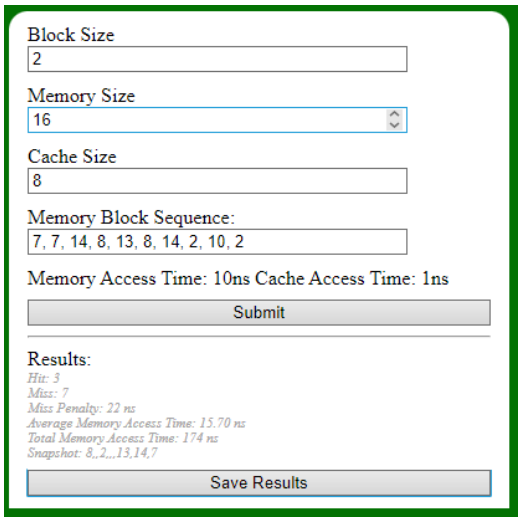
Average memory access time = hit ratio * Cache access time + miss ratio * miss penalty

Total memory access time = (hits * block size * cache access time) + [misses * block size * (memory access time * cache access time)] + (misses * cache access time)

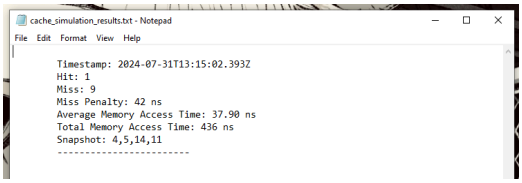
3. Simulation test cases

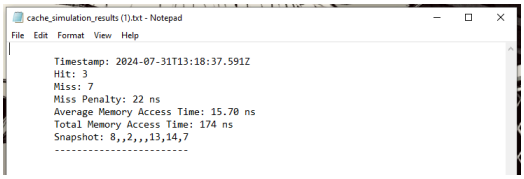
Normal Case



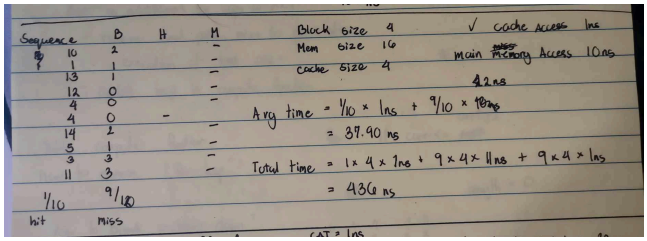


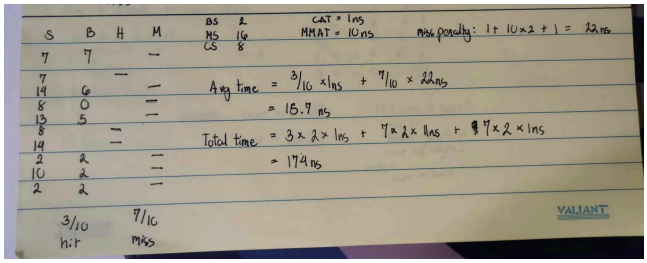
Text file outputs



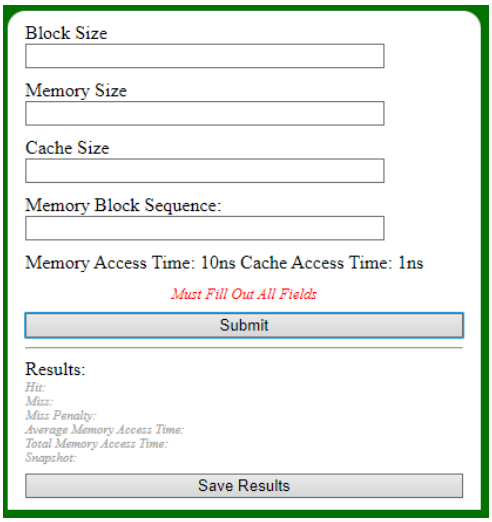


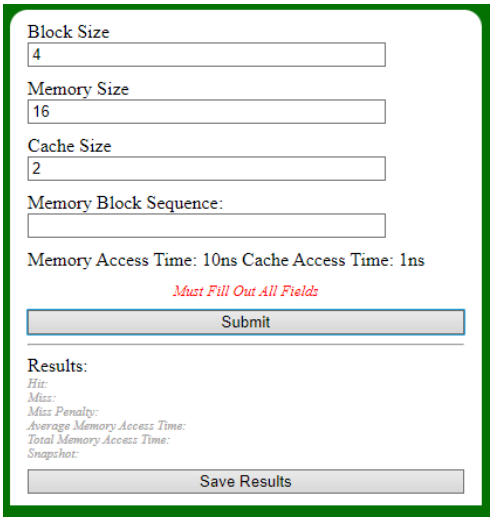
Solutions

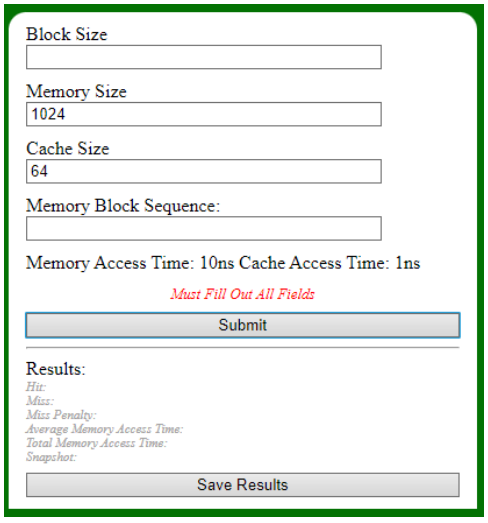




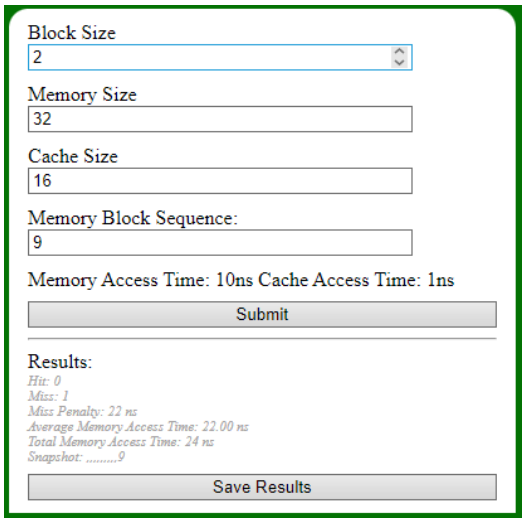
Missing fields



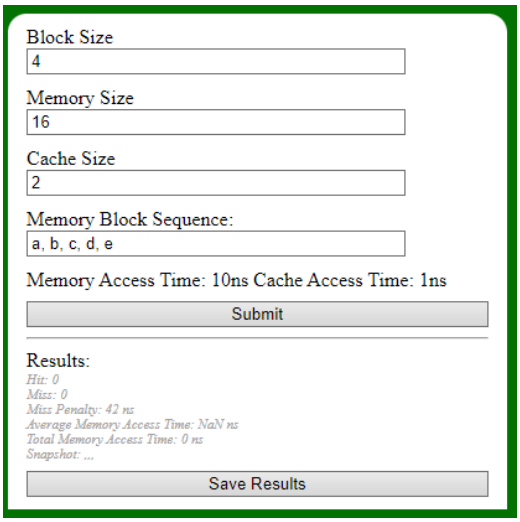


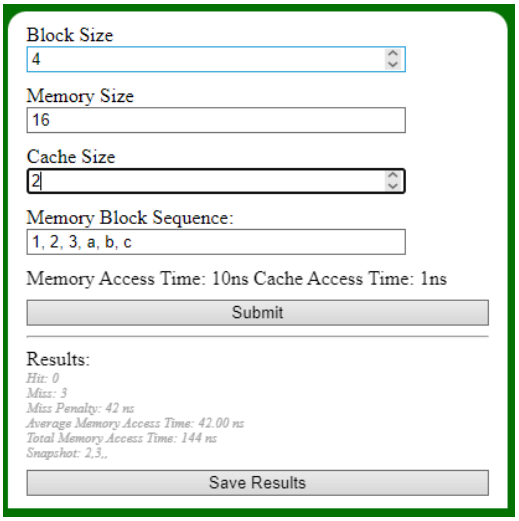


1 input in sequence



Characters in sequence





4. References

https://www.merriam-webster.com/dictionary/cache
https://www.studysmarter.co.uk/explanations/computer-science/computer-organisation-and-architecture/von-neumann-architecture/#:~:text=Von%20Neumann%20Architecture%20refers%20to,in%20the%20mid-20th%20century.
https://www.britannica.com/technology/cache-memory
https://www.baeldung.com/cs/cache-direct-mapping
https://www.geeksforgeeks.org/cache-mapping-techniques/