**Cache Simulator with Block Set Associative Mapping and MRU Replacement**

Caching is an important part of computational processing; it entails storing data in a cache for future requests. This project aims to simulate a cache with block-set associative mapping and most-recently-used replacement, with a single selector for inputting cache memory size, main memory size, and program flow in blocks or words.

The main algorithm iterates on each integer in the program flow and performs a series of three successive checks:

1. Check if the value to be inserted is already in the cache. If the value is in the cache, skip to step 4; otherwise, proceed to step 2.
2. Increment the miss counter. Check if the cache is full or does not contain any empty indexes (represented in the code by a sentinel value of -1). If the cache is not yet full, insert the current value into the empty index and skip to step 4; otherwise, proceed to step 3.
3. Check for the most recently used value within the set. Search for that value and replace it with the current value to be inserted.
4. The currently inserted value is now the most recently used value.

While the block and word modes are similar, they differ in the size of the “cache” array variable and the looping constraints, looping over a maximum of (block size) \* (set size) cells for word mode, while block mode loops over a maximum of (set size) cells.

Different formulas are used to calculate the output values:

* Miss penalty is calculated by (cache cycle time + memory cycle time) \* 2.
* Average access time is calculated by (hit % \* cache cycle time) + (miss % \* penalty time)
* The total access time is calculated by (2 \* hit count \* cache cycle time) + (2 \* miss count \* (cache cycle time + memory cycle time)) + miss count \* cache cycle time

Screenshots:

