CSARCH2 S13	1st Term 2023-2024
Cache Simulation Project	Prof. RLUy

Design a cache simulation system and analyze the various test set scenarios of the assigned cache mapping and replacement policy.

General Directions:

- Application platform: Stand-alone or web-based. Regardless, there should be a GUI (graphics user interface) instead of "text "-based output.
- Programming languages: any programming languages (C, Java, assembly language, Python, etc.).
- Application repository (source code and analysis writeup): GitHub (make sure that I can access it).

Common Specifications:

1. Number of cache blocks = 16 blocks
2. Cache line = 32 words
3. Read policy: load-through

Test cases (*n* is the number of cache blocks**):**

a.) Sequential sequence: up to 2*n* cache block. Repeat the sequence four times. Example: 0,1,2,3,…,2*n*-1 {4x}

b.) Random sequence: containing 4*n* blocks.

c.) Mid-repeat blocks: Start at block 0, repeat the sequence in the middle two times up to *n*-1 blocks, after which continue up to 2*n*. Then, repeat the sequence four times. Example: if *n*=8, sequence=0, 1,2,3,4,5,6, 1,2,3,4,5,6, 7,8,9,10,11,12,13,14,15 {4x}

Output:

a.) System output:
   a. Cache memory snapshot.
      i. Option for step-by-step animated tracing or final memory snapshot
      ii. Provide a text log of the cache memory trace (regardless of whether it is a step-by-step or final memory snapshot).
   b. Output: 1. memory access count; 2. cache hit count; 3. cache miss count; 4. cache hit rate; 5. cache miss rate; 6. average memory access time; 7. total memory access time

b.) Detail analysis of the three test cases. It will be submitted as "readme" to your GitHub. Note: Don't forget to specify the full specs of your cache simulation system

c.) Video containing the "walkthrough" of your system. Specify the link, or it can be stored in GitHub.

d.) **Note**: the source code /executable program (stand-alone) or link to the web-based app, as well as the video and analysis writeup, should all be in GitHub.

e.) Project demo if needed. Either face-to-face or through Zoom.

Group #	Type of cache memory
01	FA + Random replacement algorithm (specify random function used)
02	FA + LRU
03	FA + MRU
04	FA + FIFO
05	4-way BSA + LRU
06	4-way BSA + MRU
07	4-way BSA + FIFO
08	8-way BSA + LRU
09	8-way BSA + MRU
10	Direct

Cache simulator (Full associative / LRU)

- Input: Block size, MM memory size (accept both blocks and words), cache memory size (accept both blocks and words), program flow to be simulated (accept both blocks and words), and other parameters deemed needed (example: can simulate cache problem set # 4,5,6,7)
- Output: number of cache hits, number of cache miss, miss penalty, average memory access time, total memory access time, snapshot of the cache memory. With option to output result in text file.

* Cache simulator (Block-set-associative / LRU)

- Input: Block size, set size, MM memory size (accept both blocks and words), cache memory size (accept both blocks and words), program flow to be simulated (accept both blocks and words) and other parameters deemed needed (example: can simulate cache problem set # 4,5,6,7)
- Output: number of cache hits, number of cache miss, miss penalty, average memory access time, total memory access time, snapshot of the cache memory. With option to output result in text file.

* Cache simulator (Block-set-associative / MRU)

- Input: Block size, set size, MM memory size (accept both blocks and words), cache memory size (accept both blocks and words), program flow to be simulated (accept both blocks and words) and other parameters deemed needed (example: can simulate cache problem set # 4,5,6,7)
- Output: number of cache hits, number of cache miss, miss penalty, average memory access time, total memory access time, snapshot of the cache memory. With option to output result in text file.