## Computer Systems Architecture 2022/23
## Testing your Cache Controller Simulator

---

One approach to testing your implementation of the Cache Simulator is to create memory trace files containing sequences of read / write memory accesses for which you know the expected values of the performance counters: NRA, NWA, NCRH, NCRM, NCWH and NCWM. You can also create sequences of read / write memory accesses that will test your implementation of the tag bit comparison as well as the Valid and Dirty bits.

The memory trace files are simple text files with a fixed format and so you can create them using simple text editor such as Notepad++.

Ideally, we will want to create the 20-bit CPU addresses for these trace files without needing to write another program!

If we simulate a cache memory with 256 blocks, each containing 16 words, then the process of creating 20-bit processor addresses from the Block Offset, Cache Memory Block ID and the Tag Bits is relatively straightforward. The Block Offset is a 4-bit value ($2^4$ = 16 words in a block) and maps to one hexadecimal digit. The Cache Memory Block ID is an 8-bit value ($2^8$ = 256 blocks) and maps to two hexadecimal digits. Lastly, there are the 8 Tag Bits, and these map to two hexadecimal digits.

Below is an example of how a Block Offset of 0x5, a Cache Memory Block ID of 0x83, and the Tag Bits 0x0C form the CPU address 0x0C835.

| $A_{19}$ | | | | | | | $A_{12}$ | $A_{11}$ | | | | | | | $A_4$ | $A_3$ | | | $A_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Tag Bits | | | | | | | | Cache Memory Block ID | | | | | | | | Block Offset | | | |

| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | C | | | | 8 | | | | 3 | | | | 5 | | | |

**Example 1: Validate cache hit counting**

By keeping the Cache Memory Block ID and the Tag Bits constant we can generate a sequence of cache hits. There will also be one cache miss on the initial loading of the cache block following reset. The leading zero is not specified in the address (as is the case in the bubble sort trace files) and a comment has added that will not be present in the actual trace file.

```
R  C835      ; cache miss; 16 words moved from external memory to cache memory
R  C836      ; cache hit
R  C837      ; cache hit
R  C838      ; cache hit
R  C839      ; cache hit
```

**Example 2: Validate write back of cache blocks**

By writing to a block in the cache and then triggering it being overwritten we can test the implementation of the Dirty bit logic.

```
R C835      ; cache miss; 16 words moved from external memory to cache memory
W C835      ; cache hit; sets dirty bit
R D830      ; cache miss; 16 words moved from cache memory to external memory
            ; 16 words moved from external memory to cache memory
```

If you wish to validate your program with a different block size, then you can simulate a cache memory with 16 blocks, each containing 256 words. This configuration again maps conveniently to hexadecimal address digits.