ECED 3403 Computer Architecture Assignment 2 Testing Report

Connor McLeod B00851621

https://github.com/weakbox/eced3403-emulator

Test 1: Basic Cache Test (Direct-Mapping)

Setup: The provided "cache_basic.asm" assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to direct mapping mode.

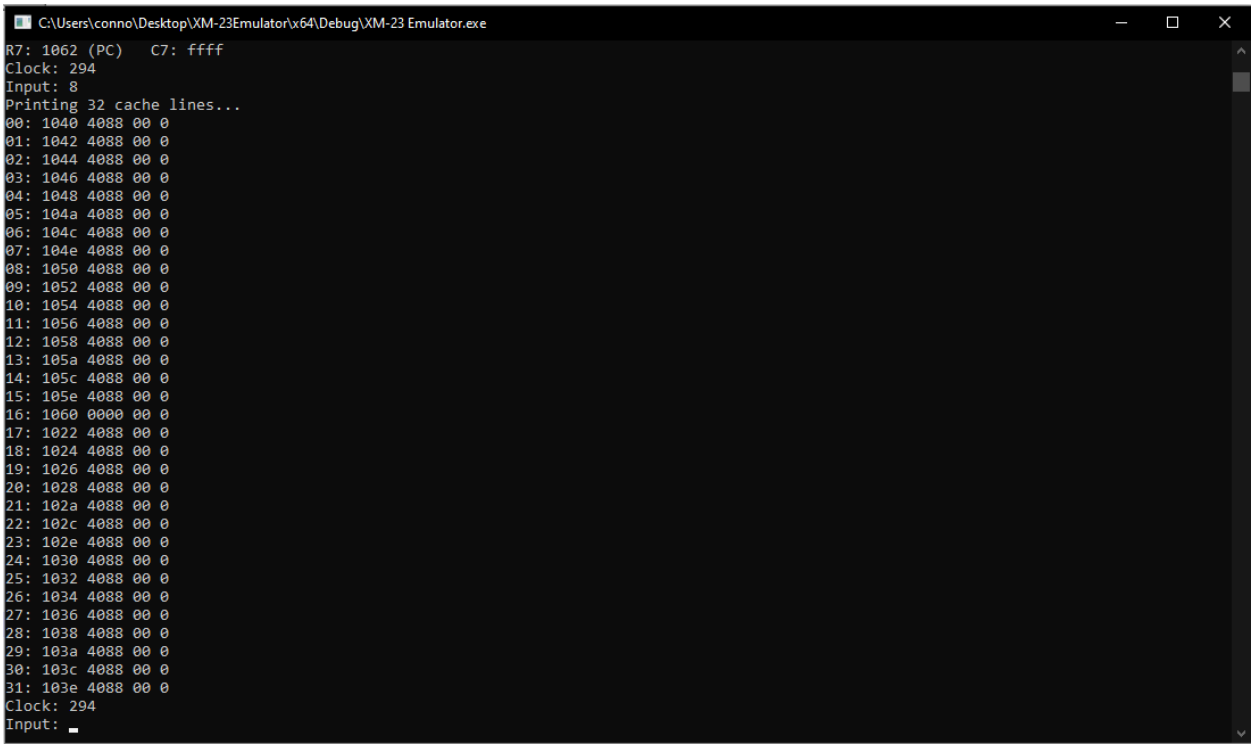
Expected Results: The cache should be completely filled with addresses from the IR. There should be no reported usage values as we are using direct mapping.

Results: The cache was correctly filled with addresses and instructions from the IR. We can see that the 49th instruction (0000 at address 1060) has been placed in cache line 16. This is correct, as from the direct mapping cache algorithm:

$$\frac{\text{address}}{2} \% 32 = \text{index}$$

$$\frac{1060}{2} \% 32 = 16$$

The index is indeed 16.



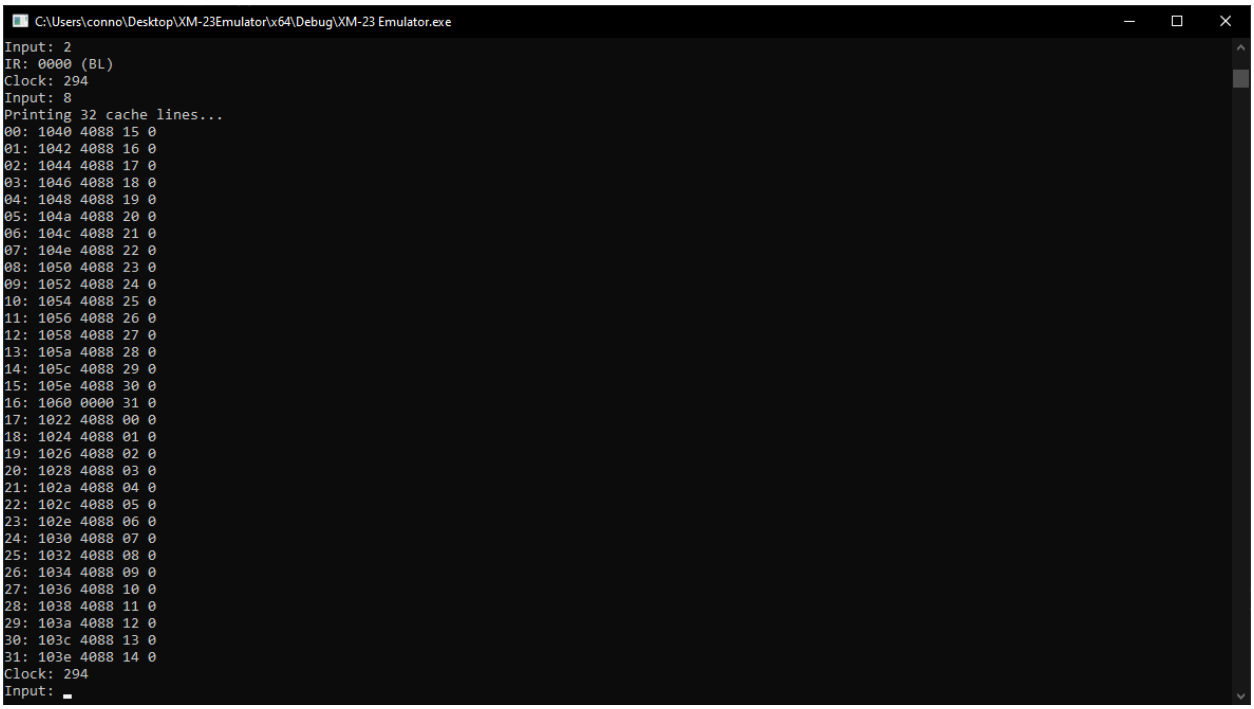
Pass/Fail: Pass. The emulator performs as expected.

Test 2: Basic Cache Test (Associative)

Setup: The provided “cache_basic.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to associative mode.

Expected Results: The cache should be completely filled with addresses from the IR. The usage values should be correctly updated as we are using associative cache organization.

Results: The cache was correctly filled with addresses and instructions from the IR. The usage values are also correctly represented. We can see that the 49th instruction (0000 at address 1060) has been placed in cache line 16. It is the most recently used cache line, represented by its usage value of 31.



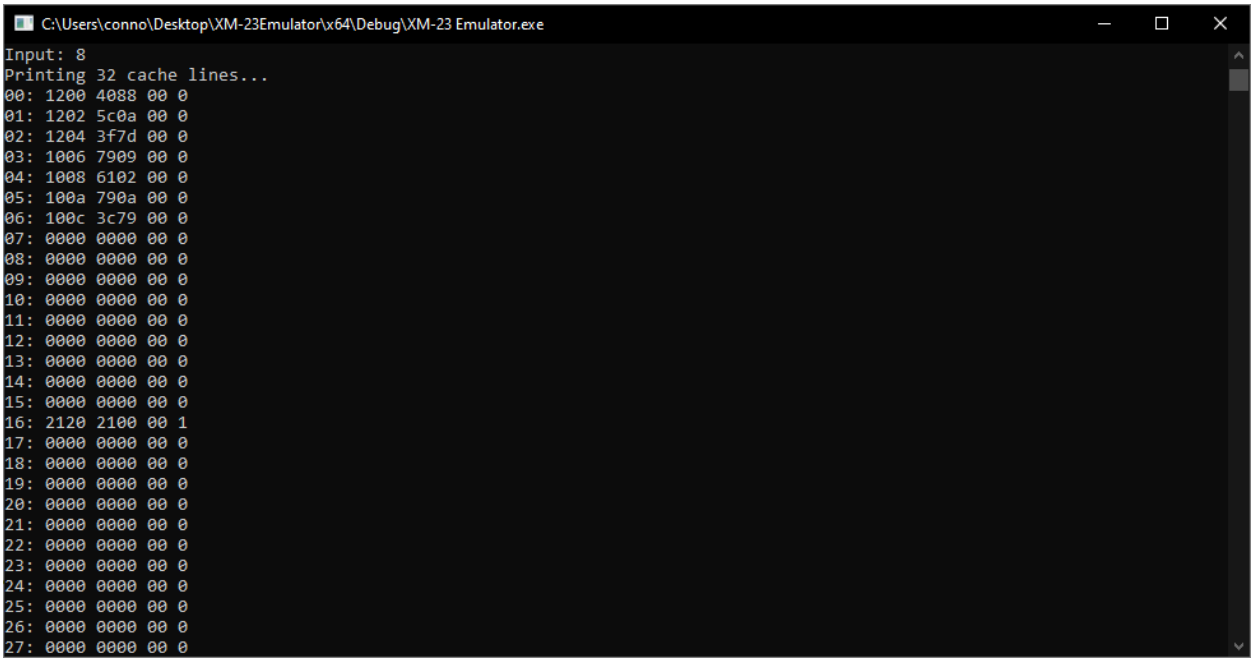
Pass/Fail: Pass. The emulator performs as expected.

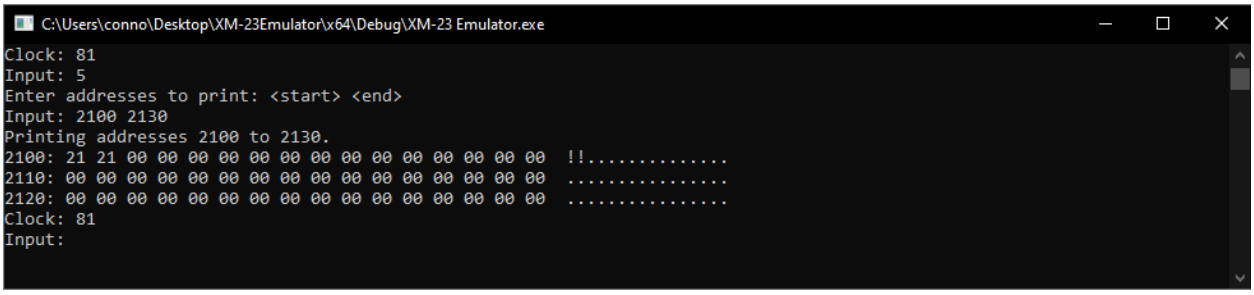
Test 3: Writing to Cache (Write-Back)

Setup: The provided “cache_write.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to direct mapping mode, using the write-back policy.

Expected Results: The contents of a cache line should only be written to memory after it has been evicted from the cache. The mechanism behind this test can be better understood by looking at the provided test code.

Results: We can see that the write performed at address #2100 has been written to main memory, as it has been evicted from the cache. However, the write performed at address #2120 has not been written to main memory, as it has not been evicted from the cache yet. We can see that the dirty bit is set for this write, indicating that when it is evicted from the cache it will be written to main memory.





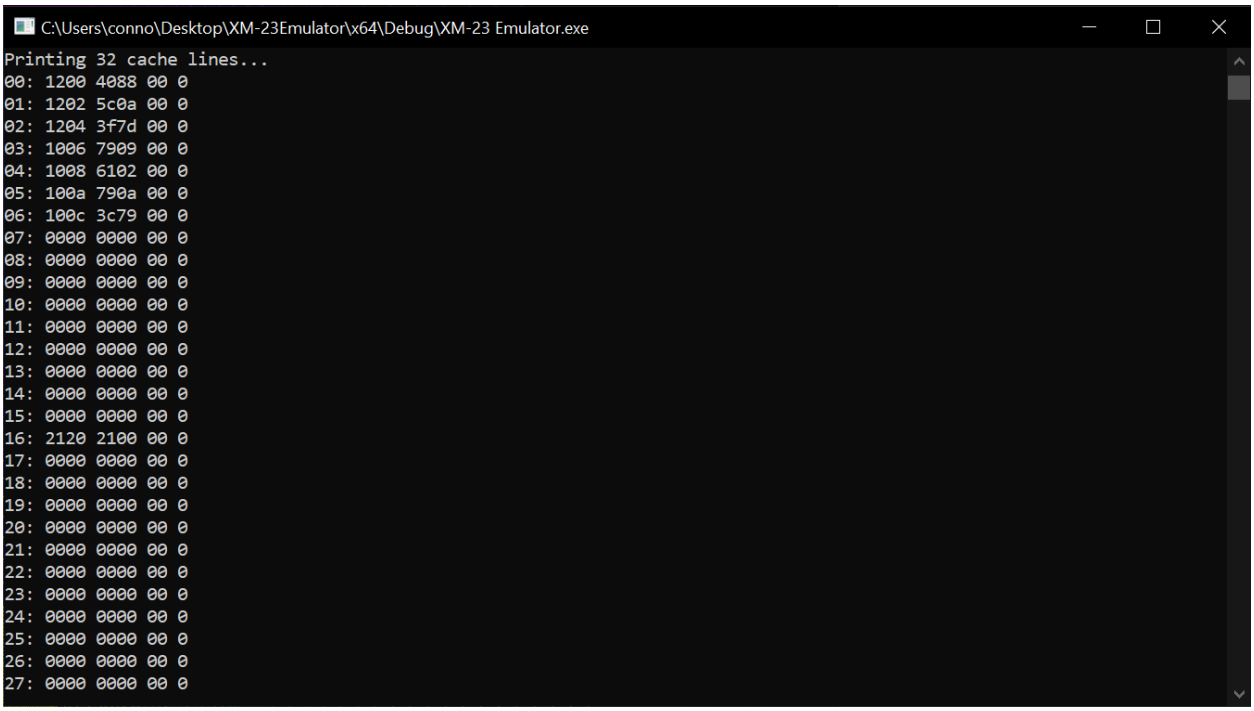
Pass/Fail: Pass. The emulator performs as expected.

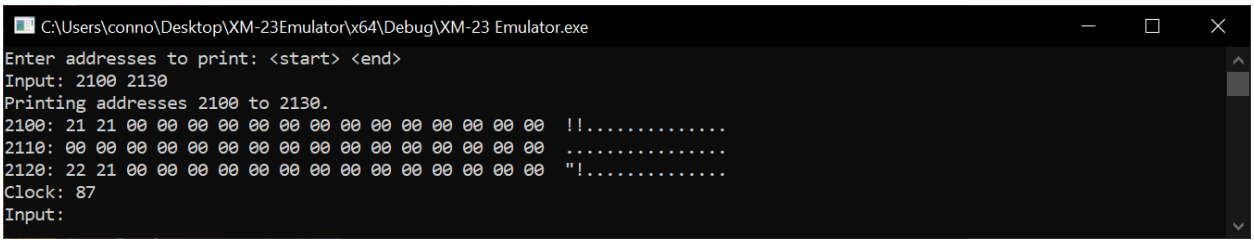
Test 4: Writing to Cache (Write-Through)

Setup: The provided “cache_write.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to direct mapping mode, using the write-through policy.

Expected Results: The contents of a cache line should be written to memory regardless of whether it has been evicted from the cache.

Results: We can see that both the writes performed at address #2100 and #2120 have been written to main memory. The dirty bit has not been set for the write performed at address #2120, as it is not necessary for the write-through policy.





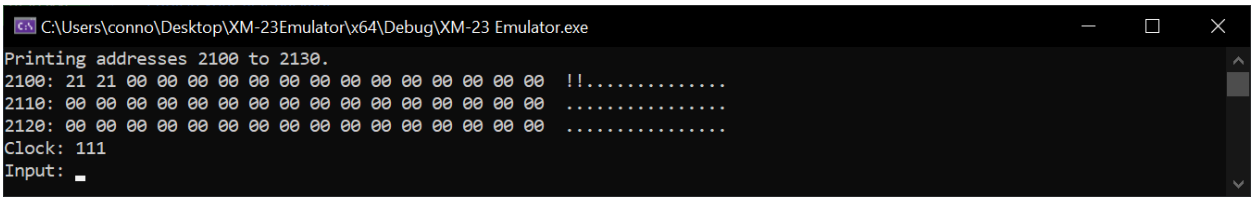
Pass/Fail: Pass. The emulator performs as expected.

Test 5: Instruction Integration

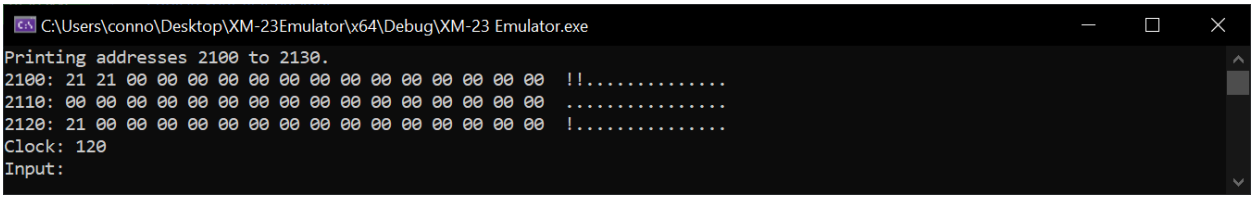
Setup: The provided “cache_inst.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to associative mode, using the write-through policy.

Expected Results: We expect to see that instructions that previously used the bus() function are able to be easily adapted to use the cache by replacing the bus() function with the cache_bus() function. These instructions should produce exactly the same results as previous testing indicated, while using the cache.

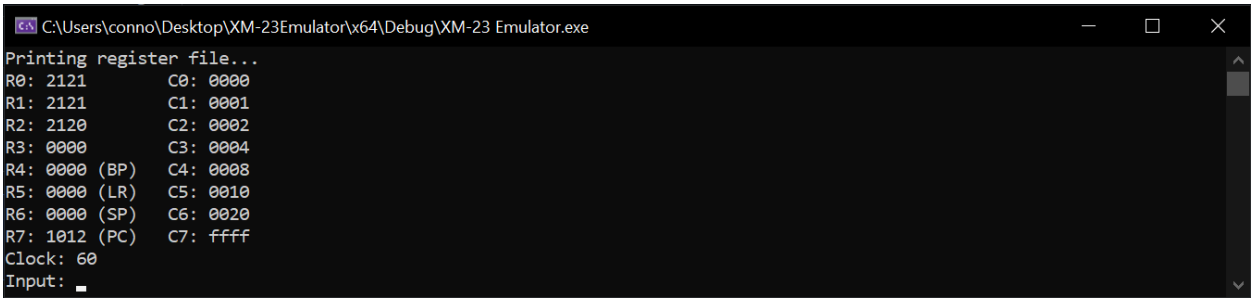
Results: The instructions provide exactly the same output that was tested previously.



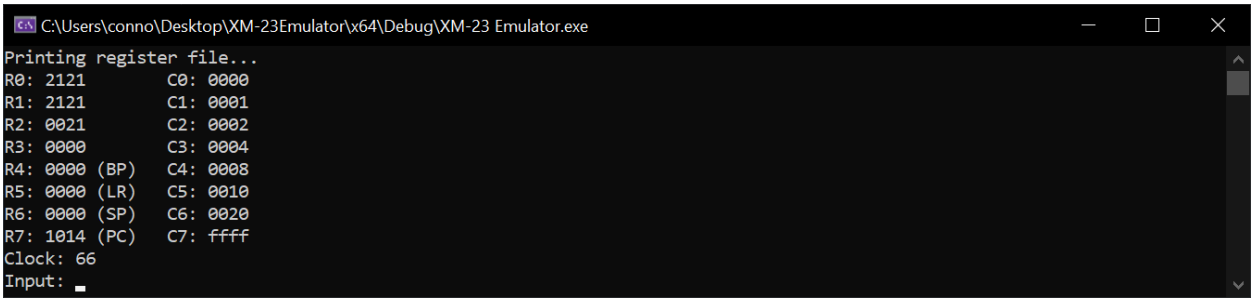
ST instruction.



ST.B instruction.



LD instruction.



LD.B instruction.

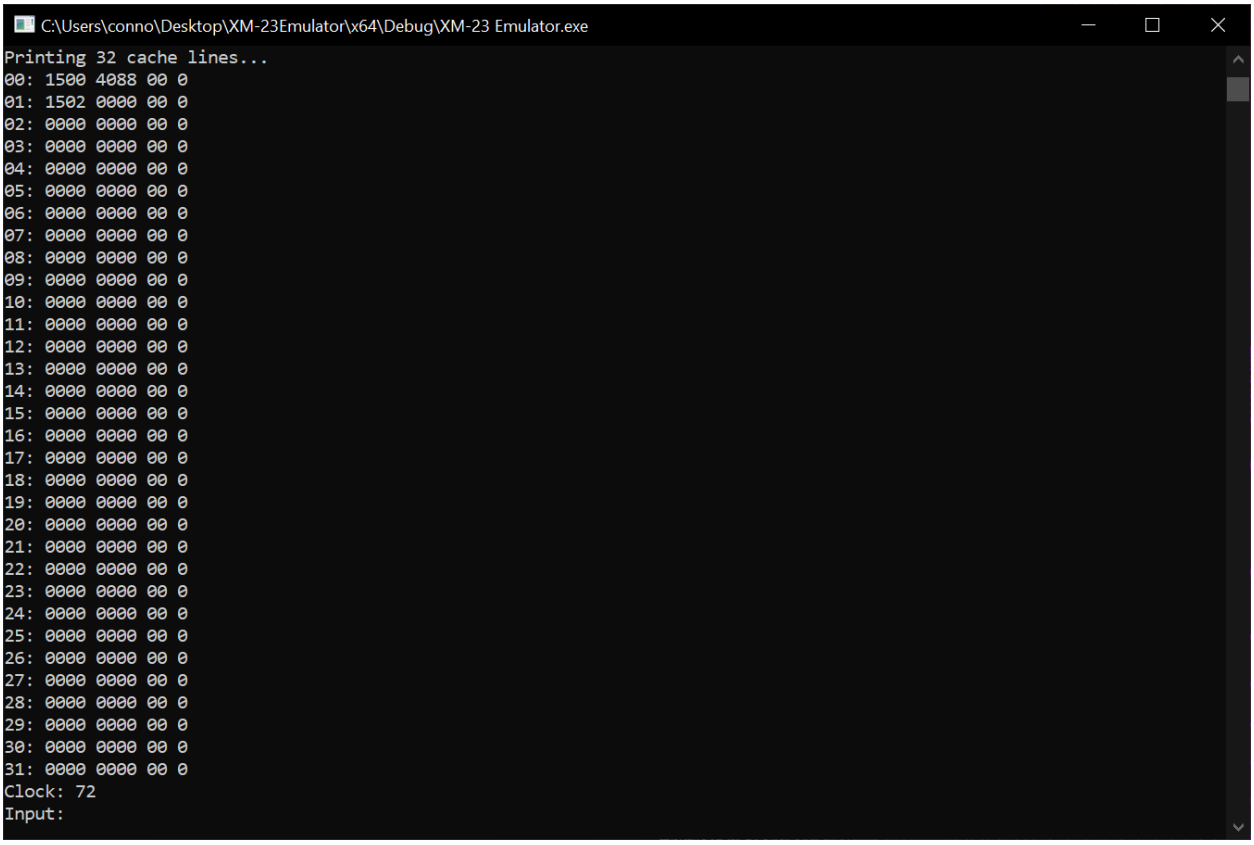
Pass/Fail: Pass. The emulator performs as expected.

Test 6: Defeating Cache (Direct-Mapping)

Setup: The provided “cache_break_dm.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to direct-mapping mode, using the write-back policy.

Expected Results: We expect to see very inefficient use of cache space due to the repeated use of similar addresses in the assembly code.

Results: Despite the code containing 11 instructions, only two of the cache lines have been populated.



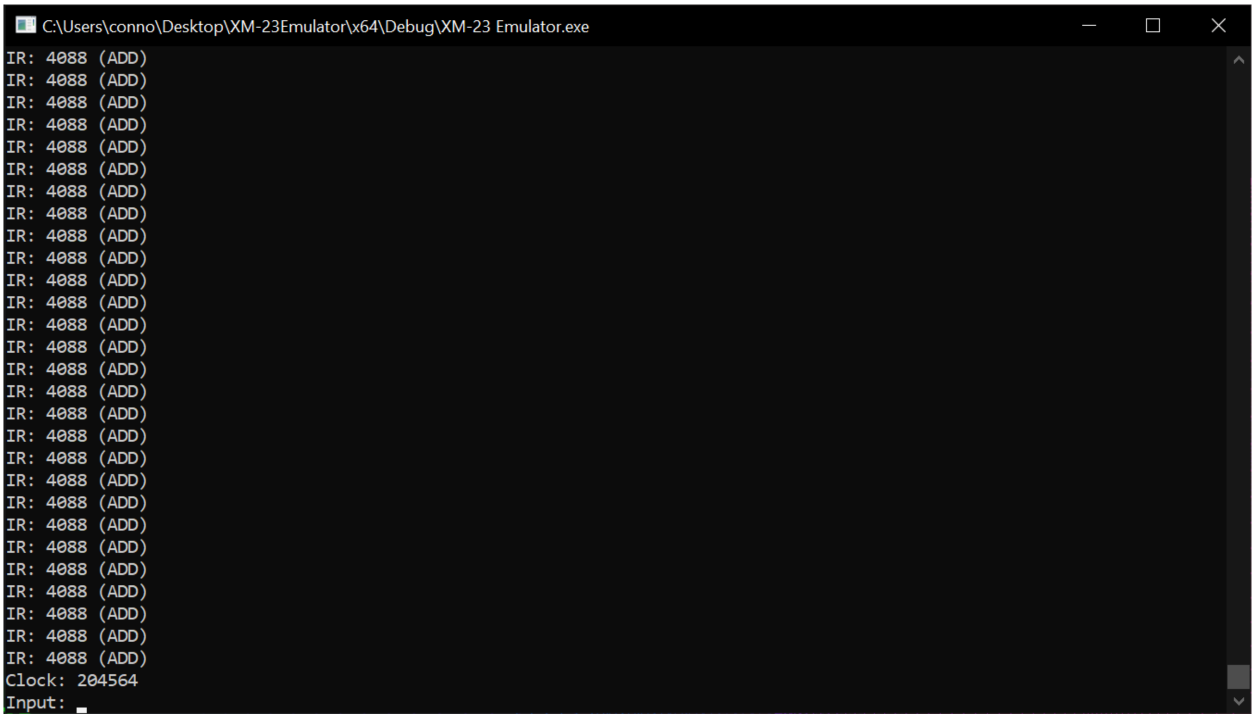
Pass/Fail: Pass. The emulator performs as expected.

Test 6: Defeating Cache (Associative)

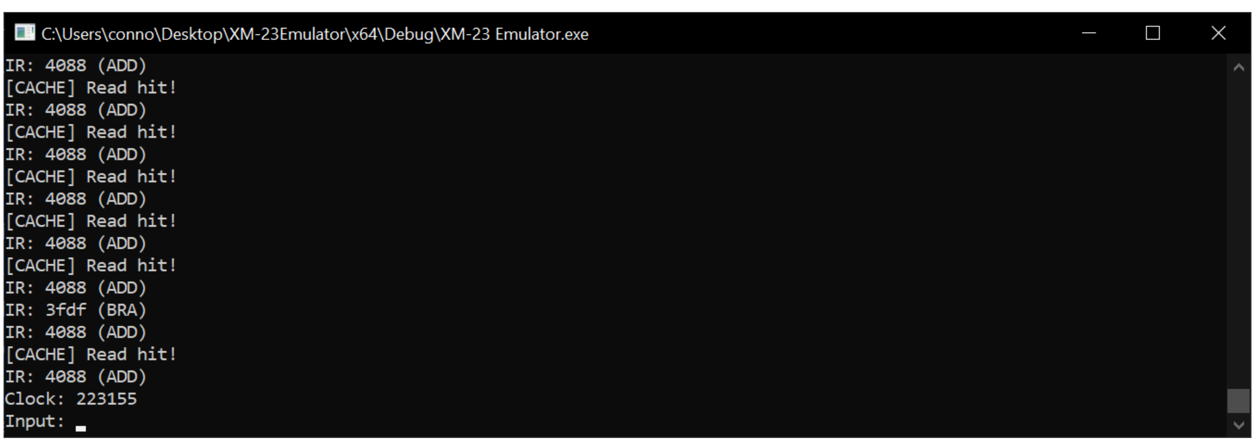
Setup: The provided “cache_break_a.asm” assembly file was assembled using the most recent version of the provided XM-23 assembler. The resulting .xme file was then run using the XM-23 emulator. The emulator cache was set to associative mode, using the write-through policy.

Expected Results: We expect to see very inefficient use of the cache, as using this organization method will not produce any hits from the cache.

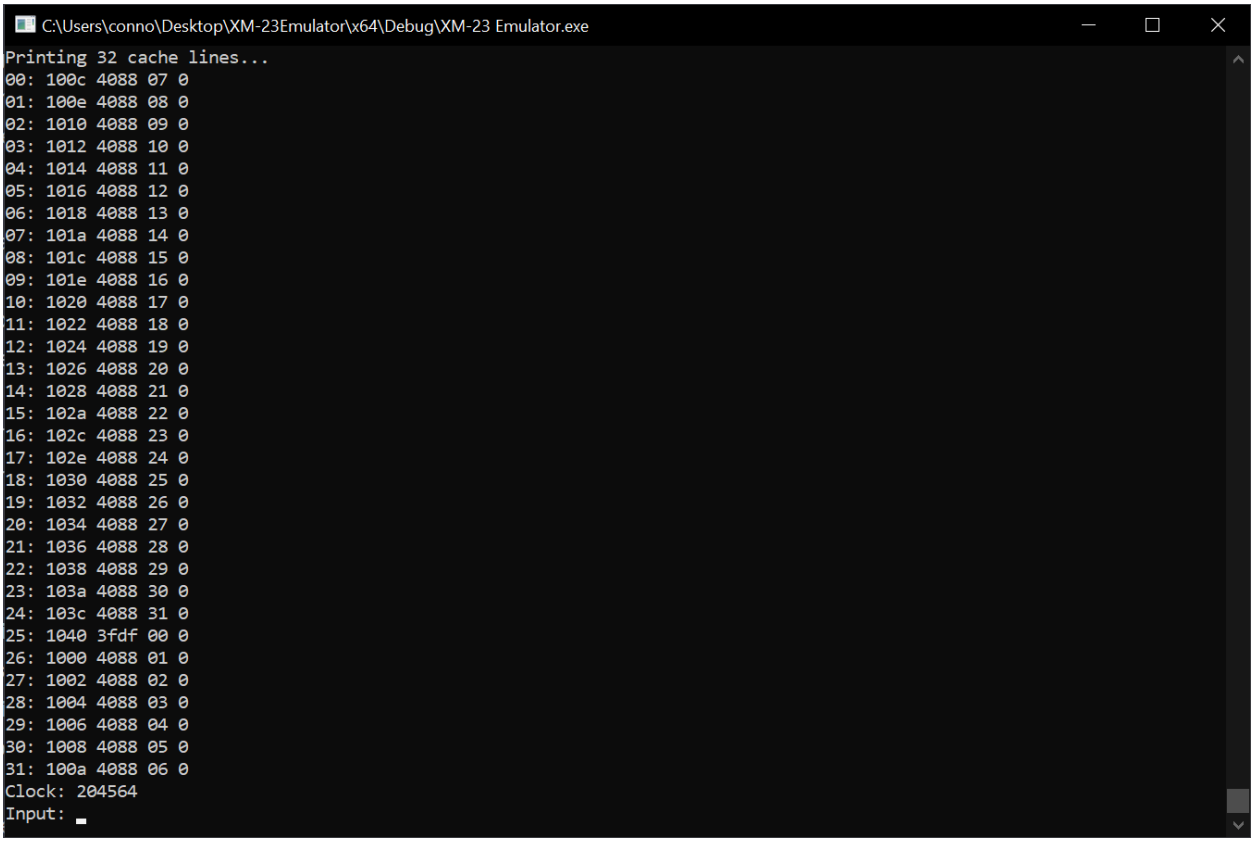
Results: With the VERBOSE flag enabled, we see that there are no hits from the cache. This does not occur when using direct-mapping, which shows us a weakness when using this cache organization method.



Associative organization.



Direct-mapping organization.



Pass/Fail: Pass. The emulator performs as expected.