SCT212-0056/2020 FAVOUR PAUL MUTURI
LAB 5

Given:

- Cache size: 16 KB = 16 × 1024 = 16,384 bytes
- Associativity: Direct-mapped
- Line size: 64 bytes
- Addressing: Physical
- Arrays X and Y:
  - Each has 4096 elements
  - Each element is 4 bytes → total size = 4096 × 4 = 16,384 bytes (16 KB) per array
- Thus, X and Y each occupy exactly one cache's worth of space.

(a) Cache misses with naive code

Memory layout and access pattern:

- X and Y are allocated consecutively in memory.
- Loop accesses one element of X, then one of Y, then modifies X.
- Access order:
  - `X[i]`: load
  - `Y[i]`: load

    - compute
    - `X[i]`: store

Cache characteristics:

- Line size = 64 bytes, so each cache line holds 64 / 4 = 16 elements.
- Cache has 16 KB / 64 = 256 lines.

Miss Types:

1. Compulsory (cold) misses:
   First access to each cache block — can't avoid.
    - X and Y each have 4096 / 16 = 256 blocks
    - Compulsory misses = 256 for X + 256 for Y = 512
2. Conflict misses:
    - Direct-mapped cache → potential conflicts if `X[i]` and `Y[i]` map to the same cache line.
    - Since both arrays are exactly 16 KB, and aligned consecutively, each `X[i]` and `Y[i]` pair map to the same cache line.
    - So:
        - Load `X[i]` → loads its block
        - Load `Y[i]` → evicts `X[i]` block
        - Store to `X[i]` → cache miss → reloads `X[i]` block
3. This means every iteration causes 3 cache misses:

    - 1 for `X[i]` load (compulsory/conflict)
    - 1 for `Y[i]` load (conflict)
    - 1 for `X[i]` store (conflict)

4. But the first 256 accesses to X and Y are compulsory. So:

Total Misses:

- X: 256 (compulsory) + 4096 − 256 = 3840 (conflict reads and writes)
- Y: 256 (compulsory) + 3840 (conflict)

Total:

- X: 256 compulsory + 3840 conflict = 4096
- Y: 256 compulsory + 3840 conflict = 4096
  → Total data cache misses = 8192

Miss rate:

Each iteration = 2 loads + 1 store = 3 memory accesses
Total memory accesses = 4096 × 3 = 12,288

Miss rate=819212288≈66.7%\text{Miss rate} = \frac{8192}{12288} ≈ 66.7\%

Answer (a):

- Compulsory misses: 512 (256 for X, 256 for Y)
- Conflict misses: 7680
- Capacity misses: 0 (arrays fit in cache, but conflict occurs)
- Total misses: 8192

- Miss rate: ~66.7%

(b) Software solution to reduce misses

Technique: Loop Interchange / Blocking

If you can't change the memory layout (X and Y still next to each other), interleaving accesses won't help. But since this is a streaming access pattern, and the conflict is caused by both X and Y mapping to the same cache line, one common solution is to interleave the computation into blocks to reuse cache lines.

Software Fix: Split-loop

A better idea here: process smaller blocks of data, e.g., tile/blocking.

```
// block size should fit in half the cache to avoid eviction
#define B 128  // 128 * 4 = 512 bytes per array = 1 KB per array per block

for (int i = 0; i < 4096; i += B) {
    for (int j = 0; j < B; j++) {
        X[i + j] = X[i + j] * Y[i + j] + C;
    }
}
```

Now:

- You work on 1 KB of X and 1 KB of Y at a time (well below cache size)
- They can coexist in the cache without evicting each other
- Still direct-mapped, but you avoid overlapping addresses

Misses:

- X: 256 blocks → 256 compulsory
- Y: 256 blocks → 256 compulsory

- No conflict or capacity misses since working set per block fits in cache

Answer (b):

- Compulsory misses: 512
- Conflict misses: 0
- Capacity misses: 0
- Total misses: 512
- Miss rate: 512 / 12288 ≈ 4.17%

(c) Hardware solution to reduce misses

Solution: Change cache associativity

- Use 2-way or 4-way set-associative cache instead of direct-mapped.
- This allows blocks from X and Y that map to the same index to coexist in the same set.

Assume: 2-way associative cache

- Now each set can hold one block from X, one from Y.

Misses:

- X: 256 compulsory misses
- Y: 256 compulsory misses
- No conflict (associativity handles it), no capacity

Answer (c):

- Compulsory misses: 512
- Conflict: 0
- Capacity: 0
- Total misses: 512
- Miss rate: 512 / 12288 ≈ 4.17%

Final Summary:

Part	Compulsory	Conflict	Capacity	Total Misses	Miss Rate
(a) Naive	512	7680	0	8192	66.7%
(b) Software Blocking	512	0	0	512	4.17%
(c) Hardware (2-way)	512	0	0	512	4.17%