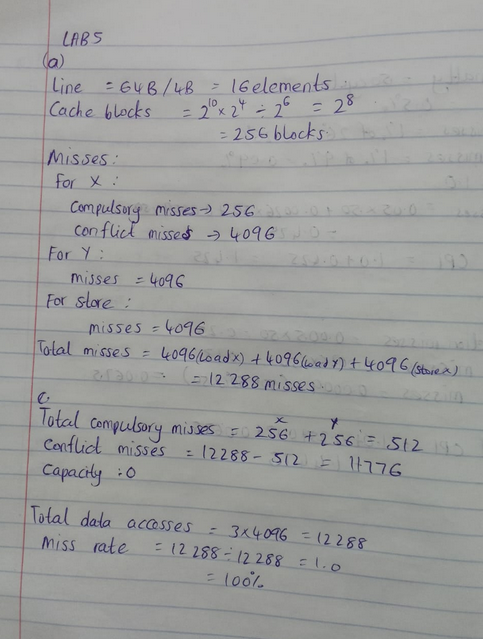
Daisy Opanga

SCT212-0052/2021

LAB 5



1. Since direct-mapped cache causes conflicts they can be avoided by reordering access or using blocking/loop tiling. Moreover, in this linear loop, an effective trick is to interleave accesses or copy one array.

Software Fix: Copy Y into a temporary array that's better aligned or process in blocks:

#define BLOCK\_SIZE 64

for (int b = 0; b < 4096; b += BLOCK\_SIZE) {

for (int i = b; i < b + BLOCK\_SIZE; i++) {

X[i] = X[i] \* Y[i] + C;

}}

Now the block size (64 elements = 256 Bytes) fits into cache without evicting each other.

Now within a block:

X block: 64 elements: 64 \* 4B = 256B: 4 cache lines

Y block: 4 lines

Each block of 64 accesses (per array) leads to 4 compulsory misses (4096/64 = 64 blocks: 64 \* 4 = 256 misses per array).

Total: Load X: 256

Load Y: 256

Store X: 256

Total = 768 misses

Compulsory misses: 512 (X + Y loads)

Conflict misses: greatly reduced, 256 (store conflicts)

Capacity: negligible (working set per block is small)

Total accesses = 3 × 4096 = 12,288

Miss rate = 768 / 12,288 = 0.0625 = 6.25%

1. Change from direct-mapped to 2-way set associative caches.

256 cache lines: 128 sets of 2 lines each

This ensures that X[i] and Y[i] mapping to same index won’t evict each other.

Compulsory misses: 4096 elements / 16 elements per line = 256 lines per array thus 512 compulsory misses.

There are no more conflict misses, because of associativity and stores to X[i] result in cache hit (value remains in cache after Y[i] access).

Compulsory misses: 512

Conflict: 0

Capacity: 0

Miss Rate: 512 / 12,288 = 4.17%