CS610: Programming for Performance
Assignment 1

Yash Gupta (190997)

1 **Problem 1:** Solution

Given, associativity = 8, cache capacity = 256 KB, block size, B = 32B, word size = 4 B, size = 32K = 2^{15}.
Number of words in a block = BL = block size / word size = 32 / 4 = 8 words/block
Number of sets = cache capacity / (associativity * block size) = 256K / (8 * 32) = 1K

Size of A = size * word size = 32K * 4 = 128 KB
As the size of array is less than the cache capacity, the whole array will fit in the cache and hence there will be no capacity or conflict misses and there will be misses only in the first iteration of the outer loop as all the elements will be in the cache from 2nd iteration onwards. Hence, we will only consider one iteration of the outer loop.

For stride = 1, As BL = 8, every 8th access will be a cold miss.
Hence total number of misses = total number of accesses / 8 = (size / stride) / 8 = (32K / 1) / 8 = 4K

For stride = 4, Every 2nd access will be a cold miss.
Hence total number of misses = total number of accesses / 2 = (size / stride) / 2 = (32K / 4) / 2 = 4K

For stride = 16, Every access will be a cold miss.
Hence total number of misses = total number of accesses = (size / stride) = (32K / 16) = 2K

For stride = 64, Every access will be a cold miss.
Hence total number of misses = total number of accesses = (size / stride) = (32K / 64) = 512

For stride = 2K, Every access will be a cold miss.
Hence total number of misses = total number of accesses = (size / stride) = (32K / 2K) = 16

For stride = 16K, Every access will be a cold miss.
Hence total number of misses = total number of accesses = (size / stride) = (32K / 16K) = 2

For stride = 32K, Every access will be a cold miss.
Hence total number of misses = total number of accesses = (size / stride) = (32K / 32K) = 1

2 **Problem 2:** Solution

Given, cache size = 64K words, number of words in a block (BL) = 8, N = 1024.

Total number of elements in a matrix = 1024 * 1024 = 1M. Hence, a row of a matrix can fit in the cache, but not the whole matrix.

As 64K / 1M = 1 / 16, at a time, only ${\frac{1}{16}}^{th}$ of a matrix can fit in the cache.

As cache size = 64K words, BL = 8, there will be a total of 64K / 8 = 8K blocks in the cache.

2.1 ikj form

For A:
No new element will be accessed as j is varied, so the multiplier corresponding to j will be 1. As k is varied, a row of A is accessed. As BL = 8, every 8th access will be a miss, hence k will have a multiplier of N/BL. As i is varied, different rows will be accessed, each resulting in the same amount of misses, hence a multiplier of N. This analysis will be the same for direct-mapped and fully-associative caches. Hence, a total of $N^2/BL = 2^{17}$ cache misses in both direct-mapped and fully-associative caches.

For B:
As j is varied, a row of B is accessed, hence a multiplier of N/BL as explained earlier. As k is varied, different columns of B are accessed, each having the same number of misses, hence a multiplier of N. As i is varied, all the misses will be repeated as the cache cannot contain the whole matrix, hence a multiplier of N. This analysis will be the same for direct-mapped and fully-associative caches. Hence, a total of $N^3/BL = 2^{27}$ cache misses in both direct-mapped and fully-associative caches.

For C:
As j is varied, a row of C is accessed, hence a multiplier of N/BL. As k is varied, the complete row of C is already in the cache, hence a multiplier of 1. As i is varied, different columns of C are accessed, each having the same number of misses, hence a multiplier of N. This analysis will be the same for direct-mapped and fully-associative caches. Hence, a total of $N^2/BL = 2^{17}$ cache misses in both direct-mapped and fully-associative caches.

Summarizing the total cache miss analysis for direct-mapped cache:

	A	B	C
i	N	N	N
i	N/BL	N	1
i	1	N/BL	N/BL
Total	N^2/BL	N^3/BL	N^2/BL

Summarizing the total cache miss analysis for fully-associative cache:

	A	B	C
i	N	N	N
i	N/BL	N	1
i	1	N/BL	N/BL
Total	N^2/BL	N^3/BL	N^2/BL

2.2 jik form

For A:
As k is varied, a row of A is accessed, hence a multiplier of N/BL. As i is varied, different columns of A are accessed, each having the same number of misses, hence a multiplier of N. As j is varied, all the misses will be repeated as the cache cannot contain the whole matrix, hence a multiplier of N. This analysis will be the same for direct-mapped and fully-associative caches. Hence, a total of $N^3/BL = 2^{27}$ cache misses in both direct-mapped and fully-associative caches.

For B:
As k is varied, different columns of B are accessed, hence a multiplier of N. This analysis will be the same for direct-mapped and fully-associative caches.

As i is varied, the same column of B will be accessed as B is not indexed by i.
For a direct-mapped cache, when a new iteration of loop i starts, only the last $\frac{1}{16}^{th}$ of the column will be in the cache as every $\frac{1}{16}^{th}$ part of the matrix maps to the same sets. Hence, all the accesses will be misses for a direct-mapped cache, hence a multiplier of N.
For a fully-associative cache, a whole column can fit in the cache as it can accomodate a total of 8K blocks. Hence, there will be no cache misses except for the first iteration, hence a multiplier of 1.

As j is varied, different columns are accessed in the loop i.
For a direct-mapped cache, every iteration will have the same number of cache misses (except for the first iteration) as the required rows of the column would have been replaced leading to conflict misses as described above, hence a multiplier of N.
For a fully-associative cache, as a whole column fits in the cache, every BL^{th} access will be a miss, hence a multiplier of N/BL.
Hence, a total of $N^3 = 2^{30}$ cache misses for a direct-mapped cache and $N^2/BL = 2^{17}$ cache misses for a fully-associative cache.

For C:
No new element will be accessed as k is varied, so the multiplier corresponding to k will be 1. As i is varied, different columns of C are accessed, hence a multiplier of N.

As j is varied, different columns are accessed in the loop i.
As in the case of B, this will result in a multiplier of N for a direct-mapped cache due to conflict misses as only $\frac{1}{16}^{th}$ of a column will remain in the cache, whereas it will result in a multiplier of N / B for a fully-associative cache.
Hence, a total of $N^2 = 2^{20}$ cache misses for a direct-mapped cache and $N^2/BL = 2^{17}$ cache misses for a fully-associative cache.

Summarizing the total cache miss analysis for direct-mapped cache:

	A	B	C
i	N	N	N
i	N	N	N
i	N/BL	N	1
Total	N^3/BL	N^3	N^2

Summarizing the total cache miss analysis for fully-associative cache:

	A	B	C
i	N	N/BL	N/BL
i	N	1	N
i	N/BL	N	1
Total	N^3/BL	N^2/BL	N^2/BL

3 **Problem 3:** Solution

Given, cache capacity = 16 MB, block size = 64 B, word size = 8 B.
Number of words in a block (BL) = block size / word size = 64 / 8 = 8 words/block.
Number of sets = cache capacity / block size = 16M / 64 = 2^{18}.
Number of words the cache can accomodate = cache capacity / word size = 16M / 8 = 2^{21}

Total number of elements in X and A = 2048 * 2048 = 2^{22}. Hence, the cache can accomodate the whole array y, a row of matrices X and A, and only half of the whole matrices X and A.

For A: As i is varied, different columns of A are accessed, hence a multiplier of N. As j is varied, different columns are accessed in the loop i. As in the previous question, this will result in a multiplier of N due to conflict misses as only half of the column will remain in the cache. As k is varied, all the misses will be repeated, hence a multiplier of N. Hence, a total of $N^3 = 2^{33}$ cache misses.

For X: No new element will be accessed as i is varied, so the multiplier corresponding to i will be 1. As j is varied, a row of A is accessed, hence a multiplier of N/BL. As k is varied, different rows will be accessed, each resulting in the same amount of misses, hence a multiplier of N. Hence, a total of $N^2/BL = 2^{19}$ cache misses.

For y: As y will fit in the cache, every BL^{th} access will be a miss the first time y is traversed, hence a total of $N/BL = 2^8$ cache misses.

4 **Problem 4:** Solution

1. For A(i, j-i) and A(i, j-i-1):
$$i = i + \Delta i$$
$$\Delta i = 0$$
$$j - i = j + \Delta j - i - \Delta i - 1$$
$$\Delta j = 1$$
Distance vector = [0, 1]. Hence flow dependence.

2. For A(i, j-i) and A(i+1, j-i):
$$i = i + \Delta i + 1$$

$$\Delta i = -1$$

$$j - i = j + \Delta j - i - \Delta i$$

$$\Delta j = -1$$

Distance vector = [-1, -1]. Hence no flow dependence.

3. For A(i, j-i) and A(i-1, i+j-1):

$$i = i + \Delta i - 1$$

$$\Delta i = 1$$

$$j - i = i + \Delta i + j + \Delta j - 1$$

$$\Delta j = 0$$

Distance vector = [1, 0]. Hence flow dependence.

There are no output dependences.