CS610: Programming for Performance
Assignment 1

Vipul Shankhpal
170803

1 Problem 1

Given cache size 256KB and line size 32B, there can be 4 words in a block and 2K no. of sets. Now, given these values, a conflict will occur when a complete column of a cache line get filled. This will happen when all the sets are occupied with 4 blocks occupied (i.e. 1 line). This gives us Nsets*lineSize no. of words that can be accommodated without a collision. Hence,

$$2K * 4 = 8K$$

. That is, A[8K] in the array will have a conflict with A[0] but due to 4 way associative, this will be accommodated. Now, when all the lines in all the sets get filled, we will have capacity misses which happens at

$$8K * 4 = 32K$$

index of the array. Now, this will not happen because the array is limited to 32K-1 words. Hence, no capacity misses.

For stride 1: Since a block can accommodate 4 words, there will be 1 cold miss in 4 accesses => 32K/4.

For the remaining strides, since stride > no. of words a block can accommodate, there will always be a cold miss. => 32K/stride.

2 Problem 2

1. Direct Mapped Cache:

$$No.of sets = cachesize/linesize = 32K/8 = 4K$$
$$blocksize = 8words$$
$$Arraysize = 512 * 512words$$

=> only 1/8 of array can fit in the cache => a[64] will evict a[0], since 64*512 words = cache size 1. ikj
A) j -> independent, k -> one cold miss in 8 => N/8, i -> new row each time, not stored in cache before => N

B) j -> one cold miss in 8 => N/8, k -> new row each time, not stored in cache before => N, i -> A[64] will evict A[0] => conflict misses => N

C) j -> one cold miss in 8 => N/8, k -> independent and entire row is stored in cache, i -> new row each time, not stored in cache before => N

2. jik
A) k -> one cold miss in 8 => N/8, i -> new row each time, not stored in cache before => N, j -> conflict misses => N

Preprint. Under review.

B) k -> new row each time, A[64] will evict A[0] => N, i -> always a miss due to eviction => N, j -> always a miss due to eviction => N

C) k -> 1 cold miss => 1, i -> new row each time, A[64] will evict A[0] => N, j -> always a miss due to eviction => N

2. Fully Associative

1. ikj
A) j -> independent, k -> one cold miss in 8 => N/8, i -> new row each time, not stored in cache before => N

B) j -> one cold miss in 8 => N/8, k -> new row each time, not stored in cache before => N, i -> A[64] will evict A[0] => conflict misses => N

C) j -> one cold miss in 8 => N/8, k -> independent and entire row is stored in cache, i -> new row each time, not stored in cache before => N

2. jik
A) k -> one cold miss in 8 => N/8, i -> new row each time, not stored in cache before => N, j -> A[64] will evict A[0], conflict misses => N

B) k -> cold misses => N, 512*8 size matrix's left part will be completely stored in cache which will lead to one cold miss in 8 for j's iterations. i -> always a hit, since k's cold misses are stored => 1

C) k -> 1 cold miss => 1, i -> new row each time => N, 512*8 size matrix's left part will be completely stored in cache causing one in 8 cold misses for j's iteration => N/8. Where N is 512.

	A	B	C
i	N	N	N
k	N/8	N	1
j	1	N/8	N/8

Table 1: Direct Mapping, ikj

	A	B	C
j	N	N	N
i	N	N	N
k	N/8	N	1

Table 2: Direct Mapping, jik

	A	B	C
j	N	N/8	N/8
i	N	1	N
k	N/8	N	1

Table 3: Fully Associative, jik

3 Problem 3

Following the same analysis in problem 2 first part, we can find that A[512] will evict A[0], since 512*4096 words = cache size.

A) i -> row wise access, always new row with A[512][] evicting A[0][] => conflict misses => N, j => always miss => N, k -> same reason => N

X) i -> independent -> 1, j -> column wise access, 1 cold miss in 8 => N/8, k -> new row => N
Note: Multiplying each column gives us the total number of misses in each table for each array.

	A	B	C
i	N	N	N
k	N/8	N	1
j	1	N/8	N/8

Table 4: Fully Associative, ikj

	A	X
k	N	N
j	N	N/8
i	N	1

Table 5: