Malay Pandey
CS 610A

Assignment I

Roll No: 22111090
August 29, 2023

Problem 1

The cache has a capacity of 256 KB, and since a single word takes 4 B (for float),
$\therefore$ Cache Size (CS) = 64 K words
Line Size (B) = 32 B = $\frac{32}{4}$ words = 8 words
Array Size (N) = 32 K words
Number of Lines (NL) = $\frac{CS}{B} = \frac{64K}{8}$ = 8K
Number of sets in 8-way set-associative cache (NS) = $\frac{NL}{8} = \frac{8K}{8}$ = 1024

For an 8-way set-associative cache, let us assume that A[0] maps to set 0. Since the line size is 8 words, so A[0], A[1], A[2], ..., A[7] would map to the same set 0. If i is a multiple of 8, A[i] would map to the set index ((i/8) mod 1024). Therefore, 8K would map to the same set as 0, similarly 2*8K would also map to the same set, and so on.

0	A[0], A[1] $\cdots$ A[7] A[8K], A[8K+1] $\cdots$ A[8K+7] A[2*8K], A[2*8K+1] $\cdots$ A[2*8K+7] A[3*8K], A[3*8K+1] $\cdots$ A[3*8K+7] The remaining 4 lines in each set would remain empty due to smaller array size
. .	$\cdots$ $\cdots$
1023	A[8K-8], A[8K-7] $\cdots$ A[8K-1] A[2*8K-8], A[2*8K-7] $\cdots$ A[2*8K-1] A[3*8K-8], A[3*8K-7] $\cdots$ A[3*8K-1] A[4*8K-8], A[4*8K-7] $\cdots$ A[4*8K-1] The remaining 4 lines in each set would remain empty due to smaller array size

Since the size of our array is smaller than the capacity of our cache, hence our entire array will fit into the cache. Therefore, we would only incur cold misses. Furthermore, since the outer loop accesses the same set of elements from our array again and again, therefore, we would incur no misses for the outer loop.

- **Stride = 1:** Here, we would incur a cold miss every 8 iterations as we are trying to access every element of the array.
  $\therefore$ Total number of cold misses = $\frac{N}{8} = \frac{32K}{8} = 4\text{K}$.

- **Stride = 4:** Here, we would incur a cold miss every alternate iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 4.
  $\therefore$ Total number of cold misses = $\frac{N}{4*2} = \frac{32K}{8} = 4\text{K}$.

- **Stride = 16:** Here, we would incur a cold miss every iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 16.
  $\therefore$ Total number of cold misses = $\frac{N}{16*1} = \frac{32K}{16} = 2\text{K}$.

- **Stride = 32:** Here, we would incur a cold miss every iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 32.
  $\therefore$ Total number of cold misses = $\frac{N}{32*1} = \frac{32K}{32} = 1\text{K}$.

- **Stride = 2K:** Here, we would incur a cold miss every iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 2K.
  $\therefore$ Total number of cold misses = $\frac{N}{2K*1} = \frac{32K}{2K*1} = 16$.

- **Stride = 8K:** Here, we would incur a cold miss every iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 8K.
  $\therefore$ Total number of cold misses = $\frac{N}{8K*1} = \frac{32K}{8K*1} = 4$.

- **Stride = 32K:** Here, we would incur a cold miss every iteration since a line can contain only 8 contiguous elements. The number of elements that we are trying to access also gets decreased by a factor of 32K.
  $\therefore$ Total number of cold misses = $\frac{N}{32K*1} = \frac{32K}{32K*1} = 1$.

Problem 2

Cache Size (CS) = 64 K words
Line Size (B) = 8 words
Array Size (N*N) = $2^{10} * 2^{10}$ words = 2^{20} words = 1024 K words
$\therefore$ Number of Lines (NL) = $\frac{CS}{B} = \frac{2^{16}}{8} = 2^{13}$ = 8K

For a direct mapped cache, let us assume that A[0][0] maps to memory location 0. Since the line size is 8 words, so A[0][0], A[0][1], A[0][2], ..., A[0][7] would map to the same memory block 0. A[0][8], A[0][9], ..., A[0][15] would map to memory block 1 and so on.

0	A[0][0], A[0][1] $\cdots$ A[0][7]
1	A[0][8], A[0][9] $\cdots$ A[0][15]
. .	$\cdots$ $\cdots$
128	A[1][0], A[1][1] $\cdots$ A[1][7]
. .	$\cdots$ $\cdots$
$2^{13} - 1$	A[63][1016], A[63][1017] $\cdots$ A[63][1023]

Each row of any array occupies 128 blocks, so, in general an element A[i][0] would occupy ((i*128) mod 8K)th memory block. Thus, in case of direct mapped cache, if we have column major access, we would face conflict misses in case we have A[0][0], ..., A[0][7] in memory and we want to load A[64][0], ..., A[64][7] into memory, since both will try to access the same memory block. Thus, in the case of column major access, only $\frac{8K}{128} = 64$ lines are utilised instead of total 8K lines, with no reuse of cache lines.

Analysis of *ikj* form:

```
for (i = 0; i < N; i++)
    for (k = 0; k < N; k++)
        for (j = 0; j < N; j++)
            C[i][j] += A[i][k] * B[k][j]
```

- **Array A:** For fixed i and k, we have complete temporal locality of reference. Hence, we would have no miss in the case of fixed i and k. For loop k, we have spatial locality of reference, so we would only have misses in the case when we have to access a different block, i.e. N/B total misses. Since an array is stored in row-major order, we would have misses for every iteration of i in both direct and fully associative caches, i.e. N total misses in outermost loop.
  $\therefore$ **Total Cache Misses** $= N * \frac{N}{B} * 1 = \frac{N^2}{B} = 128K$ *(for both cache)*

- **Array B:** For fixed i and k, we have spatial locality of reference. Hence, we would have 1 miss every B iterations, i.e. N/B total misses for the innermost loop. Since an array is stored in row-major order, so we would have 1 miss for every iteration of k for both direct and fully associative caches, i.e. N total misses. For the outermost loop, since the cache is not large enough to hold the entire array, we would face N conflict misses in both direct and fully associative cache.
  $\therefore$ **Total Cache Misses** $= N * N * \frac{N}{B} = \frac{N^3}{B} = 128M$ *(for both cache)*

- **Array C:** For fixed i and k, we have spatial locality of reference. Hence, we would have 1 miss every B iterations, i.e. N/B total misses for the innermost loop. With each iteration of k, the same row would repeatedly be accessed in cache for both direct and fully associative cache. For every iteration of i, since we store an array in row-major order, we would have 1 miss per iteration for either cache.
  $\therefore$ **Total Cache Misses** $= N * 1 * \frac{N}{B} = \frac{N^2}{B} = 128K$ *(for both cache)*

	A	**B**	**C**
i	N	N	N
k	N/B	N	1
j	1	N/B	N/B
Total	N^2/B	N^3/B	N^2/B

direct-mapped cache

	A	**B**	**C**
i	N	N	N
k	N/B	N	1
j	1	N/B	N/B
Total	N^2/B	N^3/B	N^2/B

fully associative cache

Analysis of *jik* form:

```
for (j = 0; j < N; j++)
    for (i = 0; i < N; i++)
        for (k = 0; k < N; k++)
            C[i][j] += A[i][k] * B[k][j]
```

- **Array A:** For fixed j and i, we have spatial locality of reference, so we would have 1 miss every B^{th} iteration of the k loop, i.e. N/B misses in total. we would have 1 miss per iteration of i, since the array is stored in row-major order, i.e. N misses in total. Now, since the array size is greater than the size of cache, therefore we would not be able to utilize temporal reuse, therefore, we would have N misses for the outermost loop for either cache.
  $\therefore$ **Total Cache Misses** $= N * N * \frac{N}{B} = \frac{N^3}{B} = 128M$ *(for both cache)*

- **Array B:** For the innermost loop, we are accessing the elements in row-major order, therefore we would have N total misses for either cache. For the i loop, we don't have any index for B. This would lead to conflict misses in the case of a direct mapped cache (since all the lines would not be utilised in this case), but we would incur no miss for fully associative cache (except the compulsory cold miss for i=0). For the outermost loop j, we have column-major order access for every iteration, but since there is no spatial reuse in direct mapped cache, due to conflict misses

in the inner loops, we would have N misses in the outermost loop as well. For fully associative cache, we would incur a conflict miss every B iterations, leading to N/B misses in total for the outer loop

$$\therefore \textbf{Total Cache Misses} = N * N * N = N^3 = 1G \qquad \textit{(for direct mapped cache)}$$
$$= \frac{N}{B} * 1 * N = \frac{N^2}{B} = 128K \qquad \textit{(for fully associative cache)}$$

- **Array C:** For fixed j and i, we have complete temporal locality of reference. For loop i, we have row-major access of elements, therefore we would have 1 miss per iteration of i, i.e. N misses in total. For the outermost loop j, we have column-major order access for every iteration, but since we have no spatial reuse of space due limited cache size (and utilization), we would face 1 conflict miss at every iteration of j, i.e. N total misses for direct mapped cache. For fully associative cache, we would face conflict misses every B iterations since the spatial reuse is optimal.

$$\therefore \textbf{Total Cache Misses} = N * N * 1 = N^2 = 1M \qquad \textit{(for direct mapped cache)}$$
$$= \frac{N}{B} * N * 1 = \frac{N^2}{B} = 128K \qquad \textit{(for fully associative cache)}$$

	A	**B**	**C**
j	N	N	N
i	N	N	N
k	N/B	N	1
Total	N^3/B	N^3	N^2

direct-mapped cache

	A	**B**	**C**
j	N	N/B	N/B
i	N	1	N
k	N/B	N	1
Total	N^3/B	N^2/B	N^2/B

fully associative cache

Problem 3

The cache has capacity of 16 MB, and since a single word takes 8 B.
$\therefore$ Cache Size (CS) = 2 M words
Line Size (B) = $\frac{64}{8}$ words = 8 words
Size of A and X array $(N * N) = 2^{11} * 2^{11}$ words = 2^{22} words = 4 M words
Size of y array $(N) = 2^{11}$ words = 2 K words
$\therefore$ Number of lines (NL) = $\frac{CS}{B} = \frac{2^{21}}{8} = 2^{18} = 256K$

For direct mapped cache, let us assume that A[0][0] maps to memory location 0. Since the line size is 8 words, so A[0][0], A[0][1], A[0][2] $\cdots$ A[0][7] would map to the same memory block 0. A[0][8], A[0][9] $\cdots$ A[0][15] would map to memory block 1 and so on.

0	A[0][0], A[0][1] $\cdots$ A[0][7]
1	A[0][8], A[0][9] $\cdots$ A[0][15]
. .	$\cdots$ $\cdots$
$2^{11} - 1$	A[0][2040], A[0][2041] $\cdots$ A[0][2047]
2^{11}	A[1][0], A[1][1] $\cdots$ A[1][7]
. .	$\cdots$ $\cdots$
$2^{18} - 1$	A[127][2040], A[127][2041] $\cdots$ A[127][2047]

Each row of array A (as well as X) occupies 2K blocks, so, in general an element A[i][0] would occupy ((i*2K) mod 256K)th memory block. Thus, in case of direct mapped cache, if we have column major access, we would face conflict misses in case we have A[0][0] $\cdots$ A[0][7] in memory and we want to load A[128][0] $\cdots$ A[128][7] into memory, since both will try to access the same memory block. Thus, in the case of column-major access, only $\frac{256K}{128} = 2K$ lines are utilised instead of total 256K lines, with no reuse of cache lines. For 1-D array y, since the array size is very small compared to the cache size, therefore, we would incur only cold misses every B iteration for the innermost i loop.

- **Array A:** For fixed k and j, we would incur 1 miss per iteration since we are accessing a new row on every iteration, i.e. N total misses. Loop j accesses the elements of A in column major order, but since we have array size greater than the cache size, we would face conflict miss at every iteration of j, i.e. N total misses. Similarly, we would incur N misses for the k loop as well due to large array size.
  $\therefore$ **Total Cache Misses** $= N * N * N = N^3 = 8G$

- **Array X:** For fixed k and j, we have complete temporal locality of reference, so we would have no misses for the innermost loop. Since we are accessing elements in column-major order in the loop

j, we would have 1 miss every B^{th} iteration, i.e. N/B misses in total. For the outermost loop, we would have a miss per iteration since we are accessing a new row every iteration, i.e. N total misses.
$\therefore$ **Total Cache Misses** $= N * \frac{N}{B} * 1 = \frac{N^2}{B} = 512K$

- **Array y:** Since the array size is significantly smaller than the cache size, we would incur a cold miss every B iterations for the innermost loop i, and since the whole array is now stored in the cache, we would incur no miss for k and j.
$\therefore$ **Total Cache Misses** $= 1 * 1 * \frac{N}{B} = \frac{N}{B} = 256$

	A	**X**	**y**
k	N	N	1
j	N	N/B	1
i	N	1	N/B
Total	$N^3 = 2^{33}$	$N^2/B = 2^{19}$	$N/B = 2^8$

direct-mapped cache

Problem 4

- **Loop Dependence between A(i, j-i) and A(i, j-i-1):**
  $i = i + \Delta i \implies \Delta i = 0$
  $j - i = (j + \Delta j) - (i + \Delta i) - 1 \implies \Delta j = 1$
  Therefore, our **Distance Vector** $= (\mathbf{0}, \mathbf{1})$. Since the first non-zero value in our distance vector is positive (1), therefore, we have **true dependence** in this case.

- **Loop Dependence between A(i, j-i) and A(i+1, j-i):**
  $i = (i + \Delta i) + 1 \implies \Delta i = -1$
  $j - i = (j + \Delta j) - (i + \Delta i) \implies \Delta j = \Delta i \implies \Delta j = -1$
  Our Distance Vector comes out to be = (-1, -1), which is wrong. Hence, the source and sink must be swapped. Therefore, we get the modified **Distance Vector** $= (\mathbf{1}, \mathbf{1})$. Since, the first non-zero value in our distance vector is negative (-1), therefore, we have **anti dependence** in this case.

- **Loop Dependence between A(i, j-i) and A(i-1, i+j-1):**
  $i = (i + \Delta i) - 1 \implies \Delta i = 1$
  $j - i = (i + \Delta i) + (j + \Delta j) - 1 \implies \Delta j = -2i$
  Therefore, our Distance Vector = (1, -2). Upon loop unrolling for a few iterations, we see:

i	**j**	**R**	**W**
1	2	A[0][2]	A[1][1]
1	3	A[0][3]	A[1][2]
1	4	A[0][4]	A[1][3]
1	5	A[0][5]	**A[1][4]** †
1	6	A[0][6]	**A[1][5]** †
2	3	**A[1][4]** †	A[2][1]
2	4	**A[1][5]** †	A[2][2]

  Here, we can see that there is a true dependence between iteration (i=1, j=5) & (i=2, j=3), and (i=1, j=6) & (i=2, j=4) respectively, thus confirming our **Distance Vector = (1, -2)**. Since, the first non-zero value in our distance vector is positive (1), therefore, we have **true flow dependence**.