Assignment 1

Programming for Performance, Group SELF

Naman Singla 200619 nsingla20@iitk.ac.in

Question 1

Problem 1 **[20 marks]**

Consider the following loop.

```
float s = 0.0, A[size];
int i, it, stride;
for (it = 0; it < 100 * stride; it++) {
  for (i = 0; i < size; i += stride) {
    s += A[i];
  }
}
```

Assume an 8-way set-associative cache with a capacity of 256 KB, line size of 32 B, and word size of 4 B (for `float`). The cache is empty before execution and uses an LRU replacement policy. Given `size`=32K, determine the total number of cache misses on `A` for the following access strides: 1, 4, 16, 64, 2K, 16K, and 32K. Consider all the three kinds of misses: cold, capacity, and conflict.

Answer: NOTE: 1K = 1024
Cache Capacity = 256KB, 8-way set associative
Cache Size = $\frac{256KB}{8}$ = 32KB
Line Size = 32B, Word Size = 4B $\implies$ Number of word in a line = $\frac{32}{4}$ = 8 words
Number of sets = $\frac{256KB}{32B\times 8}$ = 1024 sets = 1K sets
Number of elements in array = 32K
Set number of array's i^{th} element = $\frac{i}{8}$ mod 1K

1. Stride = 1

Array Indices	Misses	Cache Set Number
0 – 7	1	0
8 – 15	1	1
	⋮	
8K-8 – 8K-1	1	1K-1
8K – 8K+7	1	0

So for indices in 0 – 8K-1, total misses = 1K. Since we have 8-way associative cache, we won't encounter conflict miss or capacity miss till $(8K \times 8) = 64^{th}$ array element. However, the array has only 32K elements. Hence, a similar pattern follows four more times.

Cold Misses	Capacity Misses	Conflict Misses
4K	0	0

After one iteration, the complete array is in the cache. Hence, no miss for $it > 0$.

2. Stride = 4

Array Indices	Misses	Cache Set Number
0, 4	1	0
8, 12	1	1
	⋮	
8K-8, 8K-4	1	1K-1
8K, 8K+4	1	0

This will continue till 32K; after one iteration, the complete array is In the cache. Hence,

Cold Misses	Capacity Misses	Conflict Misses
4K	0	0

3. Stride = 16

Array Indices	Misses	Cache Set Number
0	1	0
16	1	1
	⋮	
8K-16	1	1K-1
8K	1	0

This will continue till 32K; after one iteration, the complete array is In the cache. Hence,

Cold Misses	Capacity Misses	Conflict Misses
2K	0	0

4. Stride = 64, 2K, 16K, 32K
The above pattern will be followed if the stride divides 32K. Hence,

Stride	1	4	16	64	2K	16K	32K
Cold Misses	4K	4K	2K	512	16	2	1

Capacity and Conflict misses will be zero in all cases.

Question 2

Problem 2

[60 marks]

Consider a cache of size 64K words and lines of size 8 words. The matrix dimensions are 1024×1024. Perform cache miss analysis for the *ikj* and the *jik* forms of matrix multiplication (shown below) considering direct-mapped and fully-associative caches. The arrays are stored in row-major order. To simplify the analysis, ignore misses from cross-interference between elements of different arrays (i.e., perform the analysis for each array, ignoring accesses to the other arrays).

Listing 1: 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];
```

Listing 2: 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];
```

Your solution should have a table to summarize the total cache miss analysis for each loop nest variant and cache configuration, so there will be four tables in all. Justify your computations.

Answer:
Cache Size = 64K words
Line Size = 8 words
$\Longrightarrow$ Number of lines = $\frac{64K}{8}$ = 8K lines

Let's see if we can accommodate a complete column in the directly mapped cache. Number of sets = 8K Let $[i, j]$ element of the array get set number 0. Then,

$$\frac{i \times 1024 + j}{8} = 8K \times n$$

$$i \times 1024 + j = 2^{16} \times n$$, i,j<1024 Assume j divides 8

$$i = 2^6 \times n$$

This means (i,j) and (i+64,j) will map to the same cache block, given the cache is directly mapped. Hence, we cannot store a single complete column in the directly mapped cache.

Also, note that the number of misses will remain the same in row-wise access of the array. The only difference will come in column-wise access.

Now, let's solve case-wise

1. **ikj form**
   All three arrays (A, B, C) are accessed row-wise; hence, the result will remain the same for direct mapped and full associative cache.

   (a) For A
   j : No Relation (A[i][k]) $\implies$ 1
   k : Row-wise (A[i][*]) $\implies \frac{N}{B} \because$ we fetch B words at a time
   i : Repeat inner (new rows) $\implies$ N

   (b) For B
   j : Row-wise (B[k][*]) $\implies \frac{N}{B} \because$ we fetch B words at a time
   k : Repeat inner (new rows) $\implies$ N
   i : Repeat inner loops (can't hold entire array) $\implies$ N

   (c) For C
   j : Row-wise (C[i][*]) $\implies \frac{N}{B} \because$ we fetch B words at a time
   k : One row will remain in cache; all hit $\implies$ 1
   i : Repeat inner (new rows) $\implies$ N

Direct Mapped

	A	B	C
i	N	N	N
k	$\frac{N}{B}$	N	1
j	1	$\frac{N}{B}$	$\frac{N}{B}$
Total Misses	$\frac{N^2}{B}$	$\frac{N^3}{B}$	$\frac{N^2}{B}$
Total Misses	2^{17}	2^{27}	2^{17}

Full Associative

	A	B	C
i	N	N	N
k	$\frac{N}{B}$	N	1
j	1	$\frac{N}{B}$	$\frac{N}{B}$
Total Misses	$\frac{N^2}{B}$	$\frac{N^3}{B}$	$\frac{N^2}{B}$
Total Misses	2^{17}	2^{27}	2^{17}

2. **jik form**

   (a) For A
   k : Row-wise (A[i][*]) $\implies \frac{N}{B}$ $\because$ we fetch B words at a time
   i : Repeat inner (new rows) $\implies$ N
   j : Repeat inner loops (can't hold entire array) $\implies$ N

   (b) For B
   k : B[*][j] $\implies$ Column-wise $\implies$ Different for Direct and Full associative

   - Direct Mapping
     k : All cold misses $\implies$ N
     i : Repeat; But can't fit entire column $\implies$ N
     j : One complete column can't be stored $\implies$ N
   - Full Associative
     k : All cold misses $\implies$ N
     i : Repeat; Can fit entire column $\implies$ 1
     j : One complete column can be stored $\implies \frac{N}{B}$

   (c) For C
   k : No Dependence (same access) $\implies$ 1
   i : C[*][j] $\implies$ Column-wise $\implies$ Different for Direct and Full associative

   - Direct Mapping
     i : All cold misses $\implies$ N
     j : One complete column can't be stored $\implies$ N
   - Full Associative
     i : All cold misses $\implies$ N
     j : The previous column was present (B words in a list) $\implies \frac{N}{B}$

Direct Mapped

	A	B	C
j	N	N	N
i	N	N	N
k	$\frac{N}{B}$	N	1
Total Misses	$\frac{N^3}{B}$	N^3	N^2
Total Misses	2^{27}	2^{30}	2^{20}

Full Associative

	A	B	C
j	N	$\frac{N}{B}$	$\frac{N}{B}$
i	N	1	N
k	$\frac{N}{B}$	N	1
Total Misses	$\frac{N^3}{B}$	$\frac{N^2}{B}$	$\frac{N^2}{B}$
Total Misses	2^{27}	2^{17}	2^{17}

Question 3

Problem 3 **[30 marks]**

Consider the following code.

```
#define N (2048)
double y[N], X[N][N], A[N][N];
for (k = 0; k < N; k++)
  for (j = 0; j < N; j++)
    for (i = 0; i < N; i++)
      y[i] = y[i] + A[i][j] * X[k][j];
```

Assume a direct-mapped cache of capacity 16 MB, with 64 B cache lines and a word of 8 B. Assume that there is negligible interference between the arrays A, X, and y (i.e., each array has its 16 MB cache for this question), and arrays are laid out in the row-major form.

Estimate the total number of cache misses for A, X, and y.

Answer:

Cache Size = 16MB = $2^{24}B = \frac{2^{24}}{8}$ words = 2^{21} words

Words per line = $\frac{64B}{8B}$ = 8 words

- For y
  Array Size = 2048 words = 2^{11} words
  $\implies$ Cache Size > Array Size $\implies$ Complete array can be stored in cache
  Cache Misses (kji) :
  - i : y[*]; we fetch B words at a time $\implies \frac{N}{B}$
  - j : Complete array already in the cache, all hit $\implies$ 1
  - k : Complete array already in the cache, all hit $\implies$ 1

  Total misses = $\frac{N}{B} = 2^8$

- For A
  Array Size = 2048×2048 words = 2^{22} words
  $\implies$ Cache Size < Array Size $\implies$ Only half of array can be in cache
  Cache Misses (kji) :
  - i : A[*][j]; Column-wise; all miss $\implies$ N
  - j : One complete column can't be stored $\implies$ N
  - k : Complete array can't be stored $\implies$ N

  Total misses = $N^3 = 2^{33}$

- For X
  Array Size = 2048×2048 words = 2^{22} words
  $\implies$ Cache Size < Array Size $\implies$ Only half of array can be in cache
  Cache Misses (kji) :
    - i : No Dependence $\implies$ 1
    - j : Row-wise (X[k][*]) $\implies$ $\frac{N}{B}$ $\because$ we fetch B words at a time
    - k : Repeat inner (new rows) $\implies$ N

  Total misses = $\frac{N^2}{B}$ = 2^{19}

Question 4

Problem 4

[10 marks]

Consider the following loop nest.

```
for i = 1, N-2
 for j = i+1, N
  A(i, j-i) = A(i, j-i-1) - A(i+1, j-i) + A(i-1, i+j-1)
```

List all flow, anti, and output dependences, if any, using the Delta test. Show your computation. Assume all array subscript references of array `A` are valid.

Answer:
Lets see all pair-wise possibilities of **flow dependence**

1. W:$A(i, j-i)$, R:$A(i, j-i-1)$

$$\begin{aligned} i &= i + \Delta i && \implies \Delta i = 0 \\ j - i &= j + \Delta j - i - \Delta i - 1 && \implies \Delta j = 1 \end{aligned}$$

   Distance Vector = [0,1]
   Hence, **Flow Dependence**

2. W:$A(i, j-i)$, R:$A(i+1, j-i)$

$$\begin{aligned} i &= i + \Delta i + 1 && \implies \Delta i = -1 \\ j - i &= j + \Delta j - i - \Delta i && \implies \Delta j = -1 \end{aligned}$$

   Distance Vector = [-1,-1]
   Hence, **No Flow Dependence**

3. W:$A(i, j-i)$, R:$A(i-1, i+j-1)$

$$\begin{aligned} i &= i + \Delta i - 1 && \implies \Delta i = 1 \\ j - i &= i + \Delta i + j + \Delta j - 1 && \implies \Delta j = 0 \end{aligned}$$

   Distance Vector = [1,0]
   Hence, **Flow Dependence**

Lets see all pair-wise possibilities of **anti dependence**

1. W:$A(i, j-i)$, R:$A(i, j-i-1)$

$$\begin{aligned} i + \Delta i &= i && \implies \Delta i = 0 \\ j + \Delta j - i - \Delta i &= j - i - 1 && \implies \Delta j = -1 \end{aligned}$$

   Distance Vector = [0,-1]
   Hence, **No Anti Dependence**

2. W:$A(i, j-i)$, R:$A(i+1, j-i)$

$$\begin{aligned} i + \Delta i &= i + 1 &&\implies \Delta i = 1 \\ j + \Delta j - i - \Delta i &= j - i &&\implies \Delta j = 1 \end{aligned}$$

Distance Vector = [1,1]
Hence, **Anti Dependence**

3. W:$A(i, j-i)$, R:$A(i-1, i+j-1)$

$$\begin{aligned} i + \Delta i &= i - 1 &&\implies \Delta i = -1 \\ j + \Delta j - i - \Delta i &= i + j - 1 &&\implies \Delta j = -2 \end{aligned}$$

Distance Vector = [-1,-2]
Hence, **No Anti Dependence**

Lets see all pair-wise possibilities of **output dependence**
We require 2 writes. Hence, the only possible case is as follows:
W:$A(i, j-i)$, R:$A(i, j-i)$

$$\begin{aligned} i &= i + \Delta i &&\implies \Delta i = 0 \\ j - i &= j + \Delta j - i - \Delta i &&\implies \Delta j = 0 \end{aligned}$$

Distance Vector = [0,0]
Hence, **No Output Dependence**