Programming for performance - Assignment 1

Abhay Kumar Dwivedi
22111001

August 20, 2022

1 Solution

Given and Analysis :

Size of Cache = 2^{18} Bytes = 2^{15} word
Size of Line = 2^6 = 2^3 word Bytes
Size of Word = 2^3 Bytes

Number of lines in cache = 2^{15} word / 2^3 word = 2^{12} lines
Number of sets in cache = 2^{12} word / 2^3 word = 2^9 sets
Replacement policy used = LRU

Now since double = 1 word and array element is of type double, hence one array element will take 1 word space.
Size of array = 32k = 2^{15} **elements =** 2^{15} **words**

$$Also,\ number\ of\ elements\ in\ a\ cache\ block\ = \frac{size\ of\ cache\ block}{size\ of\ 1\ element} = \frac{8\ words}{1\ word} = 8\ elements$$

Now, number of bits required to access cache = 15 bits
Number of offset bits used = 3 bits
Number of set bit used = 9 bits (as there are 2^9 sets)
Number of tag bit used = 15 - (3 + 9) = 3 bits

Case - 1 : When stride = 1

Cold Miss	Conflict Miss	Capacity Miss
2^{12}	0	0

On putting stride = 1 it is clear that all the element will be accessed exactly once linearly from left to right.
Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go in set 0 as the address of it will be 000..000 and with it, elements A[1] - A[7] will also be fetched within the same block.

Accessing A[1] - A[7] will give cache hit and when A[8] will be accessed it will be a cold miss, it will go to set 1, as its address is 000...1000 and with it, elements A[9]-A[15] will be fetched within the same block and result in cache hit when accessed.
In this way all the elements will be present in cache after one iteration of outer loop, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here the pattern is that there will be 1 cold miss in every 8 elements access, that is cold miss rate will be N / 8.

$$Here\ N = 2^{15}, Hence\ number\ of\ cold\ miss = \frac{2^{15}}{2^3} = 2^{12}$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 2 : When stride = 4

Cold Miss	Conflict Miss	Capacity Miss
2^{12}	0	0

On putting stride = 4 it is clear that starting from the very first element every 4th element will be accessed exactly once linearly from left to right.
Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go in set 0 as the address of it will be 000..000 and with it, elements A[1] - A[7] will also be fetched within the same block.
Accessing A[4] will give cache hit and when A[8] will be accessed it will be a cold miss, it will go to set 1, as its address is 000...1000 and with it, elements A[9]-A[15] will be fetched within the same block and accessing the next element which is A[12] will be a hit.
In this way all the elements will be present in cache after one iteration of outer loop, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here we are accessing only two elements per block out of which 1 is a cold miss, which means we will get every block inside the cache and each block will give us 1 cold miss. Here the pattern is that there will be 1 cold miss in each block, so the number of miss will be equal to number of block which is equal to 2^{12}.

$$Here,\ number\ of\ cold\ miss = 2^{12}$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 3 : When stride = 16

Cold Miss	Conflict Miss	Capacity Miss
2^{11}	0	0

On putting stride = 16 it is clear that starting from the very first element every 16th element will be accessed exactly once linearly from left to right.
Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go in set 0 as the address of it will be 000..000 and with it, elements A[1] - A[7] will also be fetched within the same block. Accessing A[16] will again give a cold miss, and it will go to set 2 and with it, A[17]-A[23] will also be fetched in the same block as A[16] In this way half of the elements will be present in cache after one iteration of outer loop, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here we are accessing only one element out of 16 elements i.e. 2 blocks which is a cold miss, which means we will get half of total blocks inside the cache and each block inside cache will give us one cold miss.
Here the pattern is that there will be 1 cold miss in each block present in cache (note : there are only half blocks present in the cache) , so the number of miss will be equal to half of the total number of blocks.

$$Here,\ number\ of\ cold\ miss = \frac{2^{12}}{2} = 2^{11}$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 4 : When stride = 32

Cold Miss	Conflict Miss	Capacity Miss
2^{10}	0	0

On putting stride = 32 it is clear that starting from the very first element every 32th element will be accessed exactly once linearly from left to right. Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go in set 0 as the address of it will be 000..000 and with it, elements A[1] - A[7] will also be fetched within the same block.
Accessing A[32] will again give a cold miss, and it will go to set 4 and with it, A[33]-A[39] will also be fetched in the same block as A[32]
In this way one fourth of the elements will be present in cache after one iteration of outer loop, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here we are accessing only one element out of 32 elements i.e. 4 blocks which is a cold miss, which means we will get one fourth of total blocks inside the cache and each block inside cache will give us one cold miss.
Here the pattern is that there will be 1 cold miss in each block present in cache (note : there are only one fourth fraction of total blocks present in the cache) , so the number of miss will be equal to one fourth of the total number of blocks.

$$Here,\ number\ of\ cold\ miss = \frac{2^{12}}{4} = 2^{10}$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 5 : When stride = 2k

Cold Miss	Conflict Miss	Capacity Miss
2^4	0	0

On putting stride = 2k i.e. 2^{11} it is clear that starting from the very first element every $2^{11}th$ element will be accessed exactly once linearly from left to right.
Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go to set 0. Next element which is fetched is A[2^{11}] which will go in 2^8th set and so on. Now, we will access every $2^{11}th$ element which will give us a cold miss, so in total we are going to access,

$$\frac{2^{15}}{2^{11}} = 2^4 \ elements$$

and each element is going to give us a cold miss, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here the pattern is that there will be 1 cold miss for each access, so the number of miss will be equal to number of elements accessed.

$$Here, \ number \ of \ cold \ miss = 2^4$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 6 : When stride = 8k

Cold Miss	Conflict Miss	Capacity Miss
2^2	0	0

On putting stride = 8k i.e. 2^{13} it is clear that starting from the very first element every $2^{13}th$ element will be accessed exactly once linearly from left to right.
Now when the first element i.e. A[0] will be accessed it will be a cold miss and will go to set 0.
Next element which is fetched is A[2^{13}] which will go in 0th set and so on.
Now, we will access every $2^{13}th$ element which will give us a cold miss, so in total we are going to access,

$$\frac{2^{15}}{2^{13}} = 2^2 \ elements$$

and each element is going to give us a cold miss, and after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element which were accessed in first iteration.
Here the pattern is that there will be 1 cold miss for each access, so the number of miss will be equal to number of elements accessed.

$$Here, \ number \ of \ cold \ miss = 2^2$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

Case - 7 : When stride = 32k

Cold Miss	Conflict Miss	Capacity Miss
1	0	0

On putting stride = 8k i.e. 2^{15} it is clear that after first element got accessed, adding stride will cross the size of the array and only one element will be accessed and since it is accessed for the very first time it will give us a cold miss.
And after the first iteration on the outer loop gets completed, every access will be a cache hit as in all the future iterations we are accessing the same element that is the first element which was accessed in first iteration.
Here since only the first element is accessed every time so there will be only 1 cold miss due to first time access.

$$Here,\ number\ of\ cold\ miss = 2^0 = 1$$

There are no eviction at all in this example and also cache size was enough for the array Hence there is no capacity and conflict miss.

2 Solution

Case - 1a : ikj form (Direct Mapping) :

	A	B	C
i	2^9	2^9	2^9
k	2^6	2^9	1
j	1	2^6	2^6

Array A :

For j index variable, since j is not present in any of the two index, so the value will be constant across j loop and will be stored in a register so there won't be any cache miss in this case.

For k index variable, i index variable will be fixed and k is accessing elements in the fashion it is stored in the array so caching will play a vital role here, on accessing the first element, the next 7 elements will come within the same block and we will access those as well sequentially in the k index variable iteration which will give 7 hits out of 8 i.e. 1 miss out of 8.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For i index variable, on changing it, we will access next row element which won't be there in cache already which means we will get miss on every access. Therefore number of misses will be N = 2^9.

Total misses in array A = 2^{15}
Array B :

For j index variable, on fixing i and k, B is accessing element in row major order which will give only 1 miss out of 8 contiguous elements.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For k index variable, on fixing i, array B will access elements of a column row wise which are not already present in the cache, which will give miss on every access. Therefore number of misses will be N = 2^9.

For i index variable, Since the size of cache is less than the size of array, so the later elements block will evict the starting blocks and accessing them by changing j will give a miss every time. Therefore number of misses will be N = 2^9.

Total misses in array B = 2^{24}

Array C :

For j index variable, on fixing i and k, C is accessing elements in the row major fashion which will give 1 miss out of 8 elements.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For k index variable, on fixing i, even when k is changed the same row elements will be accessed each time which would be present in the cache already which means every access is going to be a hit. Therefore number of misses will be N = 1.

For i index variable, when i will be changed, array C will access the next row to the previous one accessed which means it is going to access a different row every time i changes which produces a miss every time. Therefore number of misses will be N = 2^9.

Total misses in array C = 2^{15}

Case - 1b : ikj form (4-way set associative) :

	A	B	C
i	2^9	2^9	2^9
k	2^6	2^9	1
j	1	2^6	2^6

Array A :

For j index variable, since j is not present in any of the two index, so the value will be constant across j loop and will be stored in a register so there won't be any cache miss in this case. Therefore number of misses will be N = 1.

For k index variable, since the number of lines in a row i.e. 2^9 / 2^3 = 2^6 is less than number of sets 2^{10}, Hence even on increasing the value of k from 0 to 511, the cache can accommodate the whole row, and since it is accessing elements in the way it is stored,

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For i index variable, accessing any element on changing i and fixing k i.e. A[0][x] or A[1][x] and so on will give cache miss as it is accessing element of a particular column row wise which is not already present in the cache. Therefore number of misses will be N = 2^9

Total misses in array A = 2^{15}

Array B :

For j index variable, the cache can accommodate one whole row, and since it is accessing elements in the way it is stored, it will have 1 miss out of 8 access.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For k index variable, on fixing i, array B will access elements of a column row wise which are not already present in the cache, which will give miss on every access. Therefore number of misses will be N = 2^9

For i index variable, since on each iteration of index i the elements which are accessed are not present in the cache as they are evicted out from the later elements we will have miss on each access. Therefore number of misses will be N = 2^9

Total misses in array B = 2^{24}

Array C :

For j index variable, on fixing i and k, C is accessing elements in the row major fashion which will give 1 miss out of 8 elements.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For k index variable, on fixing i, even when k is changed the same row elements will be accessed each time which would be present in the cache already which means every access is going to be a hit. Therefore number of misses will be N = 1

For i index variable, when i will be changed, array C will access the next row to the previous one accessed which means it is going to access a different row every time i changes which produces a miss every time. Therefore number of misses will be N = 2^9

Total misses in array C = 2^{15}

Case - 2a : jik form (Direct Mapping) :

	A	B	C
j	2^9	2^9	2^9
i	2^9	2^9	2^9
k	2^6	2^9	1

Array A :

For k index variable, i and j index variable will be fixed and k is accessing elements in row major order and also array elements are stored in the array in the same order. on accessing the first element, the next 7 elements will come within the same block and we will access those only in the k index variable iteration which will give 7 hits out of 8 i.e. 1 miss out of 8.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For i index variable, accessing any element on changing i and fixing j, it will access an element but on changing it, it will access next row element which is not already present in the cache and so on, so every access is going to be a miss. Therefore number of misses will be N = 2^9

For j index variable, Since the size of cache is less than the size of array, so the later elements block will evict the starting blocks and accessing them by changing j will give a miss every time. Therefore number of misses will be N = 2^9

Total misses in array A = 2^{24}

Array B :

For k index variable, on fixing j and i, array B will access elements of different row every time, hence produce a miss every time. Therefore number of misses will be N = 2^9

For i index variable, on fixing j and changing i, it will access same column by changing row every time and cache wont be able to store as B[0][0] will be stored in line 0 will B[1][0] will be stored in line 64 and continuing the pattern we will be able to store 64 rows only then the later ones will evict the earlier ones and we will get miss every time. Therefore number of misses will be N = 2^9

For j index variable, on completing iteration of i and k and increasing j by 1, the starting blocks are evicted by later blocks and we get miss on every access. Therefore number of misses will be N = 2^9

Total misses in array B = 2^{27}

Array C :

For k index variable, on fixing i and j, even when k is changed the same element will be accessed each time which would be present in the register which means every access is going to be a hit. Therefore number of misses will be N = 1

For i index variable, when i will be changed, array C will access the next row to the previous one accessed which means it is going to access a different row every time i changes which produces a miss every time. Therefore number of misses will be N = 2^9

For j index variable, on completing iteration of i and k and increasing j by 1, the starting blocks are evicted by later blocks and we get miss on every access. Therefore number of misses will be N = 2^9

Total misses in array C = 2^{18}

Case - 2b : jik form (4-way set associative) :

	A	B	C
j	2^9	2^9	2^9
i	2^9	2^9	2^9
k	2^6	2^9	1

Array A :

For k index variable, i and j index variable will be fixed and k is accessing elements in row major order and also array elements are stored in the array in the same order which means 1 miss out of 8 as there are 2^{10} sets which is greater than lines required to store the whole one row 2^{10}.

$$Therefore\ number\ of\ misses\ will\ be\ \frac{N}{8} = \frac{2^9}{2^3} = 2^6$$

For i index variable, accessing any element on changing i and fixing j, it will access an element but on changing it, it will access next row element which is not already present in the cache and so on, so every access is going to be a miss. Therefore number of misses will be N = 2^9

For j index variable, Since the size of cache is less than the size of array, so the later elements block will evict the starting blocks and accessing them by changing j will give a miss every time. Therefore number of misses will be N = 2^9

Total misses in array A = 2^{24}

Array B :

For k index variable, on fixing j and i, array B will access elements of different row every time, hence produce a miss every time. Therefore number of misses will be N = 2^9

For i index variable, on fixing j and changing i, it will access same column by changing row every time and cache wont be able to store as B[0][0] will be stored in line 0 will B[1][0] will be stored in line 64 and continuing the pattern we will be able to store 64 rows only then the later ones will evict the earlier ones and we will get miss every time. Therefore number of misses will be N = 2^9

For j index variable, on completing iteration of i and k and increasing j by 1, the starting blocks will be evicted by later blocks and we get miss on every access. Therefore number of misses will be N = 2^9

Total misses in array B = 2^{27}

Array C :

For k index variable, on fixing i and j, even when k is changed the same element will be accessed each time which would be present in the register which means every access is going to be a hit. Therefore number of misses will be N = 1

For i index variable, when i will be changed, array C will access the next row to the previous one accessed which means it is going to access a different row every time i changes which produces a miss every time as cache size is not enough to store the whole array. Therefore number of misses will be N = 2^9

For j index variable, on completing iteration of i and k and increasing j by 1, the starting blocks are evicted by later blocks and we get miss on every access. Therefore number of misses will be N = 2^9

Total misses in array C = 2^{18}

3 Solution

	A	X
k	2^{12}	2^{12}
j	2^{12}	2^{9}
i	2^{12}	1

Array A :

For index i, if we fix j and k, changing i will mean accessing element of a column row wise which won't be there in cache already which means there will be miss in each access. Therefore number of misses will be N = 2^{12}

For index j, if we fix k, on changing j, one iteration of i will be completed and cache size is smaller than the array size so the earlier blocks will be evicted out by later blocks and we will get miss in each access. Therefore number of misses will be N = 2^{12}

For index k, same thing will happen when iteration of i and j will be completed the later block will evict earlier box and there will be miss in every access. Therefore number of misses will be N = 2^{12}

Total misses in array A = 2^{36}

Array X :

For index i, if we fix j and k, changing i will mean nothing as it is not in any index in array X and value stored in register will be used multiple times producing a hit on every access. Therefore number of misses will be N = 1

For index j, if we fix k, on changing j, we will access elements of row one by one and since one cache line can store 8 words i.e. 8 elements, on accessing first element next 7 will also come in the same cache line which means we will get 1 miss out of every 8 access. Therefore number of misses will be N = 2^{12} / 2^3 = 2^9.

For index k, on changing it, we will access next row element which won't be there in cache already which means we will get miss on every access. Therefore number of misses will be N = 2^{12}.

Total misses in array X = 2^{21}