Assignment 1

Ayush Kumar (170195)

17 September 2020

I pledge on my honor that I have not given or received any unauthorized assistance.

Solutions

1. Number of sets $= \frac{CacheSize}{BlockSize*Associativity} = \frac{2^{18}Bytes}{2^5*4Bytes} = 2^{11}$
   Number of blocks in each set (associativity) = 4
   Block size = 32 Bytes = 4 words (where each word is a **double** of 8 Bytes)

   (a) Stride = 1.
   Consecutive blocks of 4 words of array A will go into sets $0, 1, 2, 3, \ldots, 2^{11} - 1$. Since a block of 4 words is fetched from the cache during a miss, the next 3 accesses after a miss will be hits. Hence 1 miss for every 4 accesses in the inner loop. Also since total size of the array is $2^{15} * 8 = 2^{18}$ Bytes, the whole array will fit in the cache. Hence after 1 iteration of the inner loop, there are no more misses in subsequent iterations.
   Total number of cache misses $= \frac{ArraySize}{4} = \frac{2^{15}}{4} = 2^{13}$.

   (b) Stride = 4
   Since the stride is 4, only first word of each block is accessed. Hence all accesses are misses in the first iteration of the inner loop, after which all accesses are hits since cache capacity is enough for all blocks.
   Total number of cache misses $= \frac{ArraySize}{4} = \frac{2^{15}}{4} = 2^{13}$

   (c) Stride = 16
   This case is similar to the previous case, except that instead of fetching consecutive blocks, every 4th block is accessed and fetched. Again all misses in the first iteration of inner loop and then all hits in subsequent iterations.
   Total number of cache misses $= \frac{ArraySize}{16} = \frac{2^{15}}{16} = 2^{11}$

   (d) Stride = 32
   Similar to the previous case, every 8th block is accessed and fetched.
   Total number of cache misses $= \frac{ArraySize}{32} = \frac{2^{15}}{32} = 2^{10}$

   (e) Stride = 2^{11}
   Similar analysis like previous case.
   Total number of cache misses $= \frac{ArraySize}{2^{11}} = \frac{2^{15}}{2^{11}} = 16$

   (f) Stride = 2^{13}
   Total number of cache misses $= \frac{ArraySize}{2^{13}} = \frac{2^{15}}{2^{13}} = 4$

   (g) Stride = 2^{15}
   Total number of cache misses $= \frac{ArraySize}{2^{15}} = \frac{2^{15}}{2^{15}} = 1$

Variant / Array	DirectMapped		FullyAssociative	
	ikj	jik	ikj	jik
A	2^{15}	2^{24}	2^{15}	2^{24}
B	2^{24}	2^{27}	2^{24}	2^{15}
C	2^{15}	2^{18}	2^{15}	2^{15}

Table 1: Number of Cache Misses in ikj and jik variants

2. (a) **ikj - DirectMapped**
   BlockSize = 8 words

Number of sets in cache $= \frac{CacheSize}{BlockSize} = \frac{2^{15}}{8} = 2^{12}$

- C $\rightarrow$ For the innermost loop, since block size is 8 and stride is 1, hence we have 1 miss every 8 accesses ($\frac{512}{8} = 64$ misses). For $i = 0$, iterating once over the j loop fills 64 blocks in sets $0, 1, 2, \ldots, 63$. The outer loop of k does not add any more misses since all words of the current row are already in the cache. No block is fetched twice.
  Total misses $= 64 * 512 = 2^{15}$

- A $\rightarrow$ The behaviour is similar to C since the pattern of indexing is same in both cases and also the innermost loop of j in this case doesn't add any more misses due to temporal locality.
  Total misses $= 2^{15}$

- B $\rightarrow$ The behaviour is same as for C and A above, except that we have an outer loop i. Since the cache cannot fit the complete array B ($2^{15} < 512 * 512$), in each iteration of the outermost loop, all blocks will have to be fetched again from the cache.
  Total misses $= 2^{15} * 2^{9} = 2^{24}$

(b) **ikj - FullyAssociative**
BlockSize = 8 words
Number of blocks in the set $= 2^{12}$

- C $\rightarrow$ Behaviour is similar to that as in **ikj - DirectMapped**, except that here the fetched blocks will all be placed in set 0.
  Total misses $= 2^{15}$
- A $\rightarrow$ Behaviour is again same as for **ikj - DirectMapped**.
  Total misses $= 2^{15}$

- B $\rightarrow$ Again, since the whole array B cannot fit in cache ($2^{15} < 512 * 512$), the answer remains same as in the **ikj - DirectMapped** case.
  Total misses $= 2^{24}$

(c) **jik - DirectMapped**
BlockSize = 8 words
Number of sets in cache $= \frac{CacheSize}{BlockSize} = \frac{2^{15}}{8} = 2^{12}$

- C $\rightarrow$ Notice that the innermost loop k will not contribute to any misses due to temporal locality, hence we can safely ignore it. We iterate over C columnwise. Since each row contains $\frac{512}{8} = 2^6$ blocks, so in loop i, the first block of each row will be fetched and placed in sets $0, 2^6, 2 \cdot 2^6, 3 \cdot 2^6, \ldots, 63 \cdot 2^6$. When $i = 64$, the next block fetched has to be put in set 0, hence the first block of first row which was already present in set 0 will be evicted. Hence, after one complete iteration of i, the first block of last 64 rows will be present in the cache. So when the next column is accessed, the first block of first row will be fetched again. This will continue on for further columns. Therefore all accesses are a miss.
  Total misses $= 512 * 512 = 2^{18}$

- A $\rightarrow$ The analysis is identical to that for B in **ikj - DirectMapped** case since the array access pattern is exactly same.
  Total misses $= 2^{24}$

- B $\rightarrow$ We have columnwise iteration of B. Also, each column is iterated 512 times due to the loop i, all of which result in complete misses since at the end of a column iteration, only the first blocks of last 64 rows are present.
  Total misses $= 512 * 512 * 512 = 2^{27}$

(d) **jik - FullyAssociative**
BlockSize = 8 words

Number of blocks in the set = 2^{12}

- C $\rightarrow$ Again we can ignore loop k due to temporal locality of accesses. Iterating over the first column results in all misses which fetches first block of each row and fills 512 blocks of the set. Due to this we get complete hits for $j = 1, 2, \ldots, 7$. In this way, we get a complete miss for every 8th column.
  Total misses = $512 * 512/8 = 2^{15}$
- A $\rightarrow$ The analysis is identical to that of B done in **ikj - FullyAssociative** case since the array access pattern is exactly same.
  Total misses = 2^{24}

- B $\rightarrow$ Every column is iterated 512 times due to loop i which will be a complete hit since a column can fully fit in the cache. Similar to C, we again have a complete miss for every 8th column.
  Total misses = $512 * 512/8 = 2^{15}$

3. Cache capacity = 2^{24} Bytes = 2^{21} words.
   BlockSize = $\frac{64}{8} = 8$ words.
   Number of sets in cache = $\frac{2^{21}}{8} = 2^{18}$

   - A $\rightarrow$ Due to the inner loops j and i, we iterate over A columnwise. Since one row of A contains $\frac{4096}{8} = 2^9$ blocks, hence the first block of the first 512 rows goes into the sets $0, 2^9, 2 \cdot 2^9, 3 \cdot 2^9, \ldots, (2^9 - 1) \cdot 2^9$. When $i = 512$, the first block of the first row will be evicted from set 0. Hence, in one iteration of the first column, we have 4096 misses (all accesses are a miss) and at the end we are left with the first block of the last 512 rows in the cache. In this way, each column is a complete miss. The full array will be iterated column-wise 4096 times due to the outermost loop k.
     Total misses = $4096 * 4096 * 4096 = 2^{36}$

   - X $\rightarrow$ We can ignore the innermost loop i due to temporal locality. Iteration is performed row-wise and since each block contains 8 words, every 8th access is a miss.
     Total misses = $4096 * 4096/8 = 2^{21}$