CS3350 Assignment 1

Mazen Baioumy
250924925

Answers to Questions:

Answer to Q1) The function factRec has an instruction locality that is poor this is since the function involves making multiple recursive calls, this causes jumps in the program execution and leads to a poor temporal locality. Each call to factRec requires the program to access a different portion of the program to be executed this leads to poor locality and in particular spatial locality. The function of factIter on other hand has a good instruction locality this is because the function involves a for loop that iterates 5 times, leading to good temporal locality. The portion of code that is repeated leads to good spatial locality due to the for loop calling the same set of instructions. For example, since n=5 the function factIter will make five iterations of the loop and perform the same set of instructions each time, leading to good temporal and spatial locality. As a further explanation, the factRec function creates five stack frames and adding them to the stack. The function must access a different memory location each time it is called, resulting in poor locality. factIter has the good locality due to the single for loop, processor predicts the next instruction and fetch it into the cache, this is what results in good temporal and spatial locality.

Answer to Q2)

Part a)

Address	Binary	Index	Block Offset	Hit or Miss	Type of Miss
8	1000	01	00	Miss	Cold
3	0011	00	11	Miss	Cold
9	1001	01	01	Miss	Conflict
7	0111	01	11	Miss	Conflict
15	1111	11	11	Miss	Conflict
20	10100	10	00	Miss	Capacity
22	10110	10	10	Miss	Capacity
2	0010	00	10	Miss	Conflict
6	0110	01	10	Miss	Conflict
0	0000	00	00	Miss	Cold

Part b)

Set 0	Set 1
2	6
0	15
9	22

Answer to Q3)

Part A)

Average memory access time (AMAT) in this situation is calculated using formula: (Hit time) + miss penalty * miss rate

For P1:
Hit time = 1.26ns
Miss penalty = 100ns
Miss rate = 0.036

AMAT = (1.26) + 100 * 0.036 = 1.26 + 3.6. = 4.86 ns is the average memory access time for P1

For P2:
Hit time = 2.17ns
Miss penalty = 100ns
Miss rate = 0.031

AMAT = (2.17) + 100 * 0.031 = 2.17+ 3.1 = 5.27 ns is the average memory access time for P2

Part B)

Cycle time is determined by the L1 hit time and is defined as the time required to execute one instruction. For both processors, the L1 hit time determines the cycle time. The CPI stall formula is CPIstall = Ideal CPI + (access count * miss rate * miss penalty / Cycle time)

For P1:
CPIstall = 2 + (0.42 * 3.6% * 100)/1.26 = 3.2

For P2:
CPIstall = 2 + (0.42 * 3.1% * 100)/2.17 = 2.6

Therefore, P1 CPIstall is 3.2 while for P2 it is 2.6, P1 is faster than P2.

Part C)

With the addition of L2 Cache we can calculate AMAT for P1 using this formula:
L2 Hit Time + L2 Miss Rate × L2 Miss Penalty
L2 Hit time = 26.24
L2 Miss Rate = 48%
L2 Miss Penalty = 100

For P1:
L2 Hit Time + L2 Miss Rate × L2 Miss Penalty
= (26.24 + 48% * 100)
= (26.24 + 0.48 * 100)
= 74.24

AMAT = 1.26 + 3.6% * 74.24
AMAT = 3.9
Therefore, with the addition of L2 Cache the AMAT got better than before adding it.

Part D)

Cycle time is determined by the L1 hit time and is defined as the time required to execute one instruction. The L1 hit time determines the cycle time. The CPI stall formula is CPIstall = Ideal CPI + (access count * miss rate * miss penalty / Cycle time) (0.42 * 0.036 = 0.01512)

For P1:
CPIstall = 2 + (0.01512 * 74.24)/1.26
CPIstall = 2 + 0.89
CPIstall = 2 + 0.89
CPIstall = 2.89

Therefore, the CPIstall for P1 with addition of an L2 cache is 2.89

Part E)

Processor P1 with an L2 cache is faster than Processor P2. To match P2's performance, P1 would need to have an L1 miss rate higher. AMAT of P1 with (L2) is equal to AMAT of P2. AMAT = 1.26 + 5.5% * 74.24
AMAT = 5.3 and since AMAT of P2 is 5.27. We need to have miss rate higher than 5.5%

Answer to Q4)

Part A)

i) A is of size 32768, we know that each cache block = 4 words = 4 x 8 bytes = 32 bytes.
The number of integers in each cache block = 32/4 = 8 integers.

We know that initially there is no data in the cache, first iteration of loop for A [] is requested therefore, there is going to be a cold miss. We know that the size of array is. 32,768. Divide. By how much fits in one row, we need to divide. One integer size = 32 bit = 4Bytes. Cache Block size = 4 word * 8 = 32 bytes, Cache Block =. 8 ints.
Number of cache blocks of data in main memory is 32768/8 = 4096
Now the number of cache misses is equal to total number of cache access minus the non-cache miss access. We also know that the array aligns with the beginning of the cache, and since we know that the first access is a miss, we need to we move to cache block with data, then we need to receive a hit for every process until the end of the cache block is reached. That is where we must read another block again. Therefore, we miss every 16 times.
Therefore, total misses = **4096 misses**

ii) Cache miss rate = number of cache misses / total number of accesses to cache
Miss rate = 4096/32768 = 0.125
0.125 * 100 = 12.5%
iii) The types of misses are cold misses

Part B)

Int = 32/8 = 4 Bytes
Cache Block Size = 4 word * 8 = 32 Bytes
Cache Block = 8 Ints
256 * 8 = 2048 bytes in cache.
In this function there are two arrays A and B which are of size 32768. Here in each iteration whenever A is required then B is also required, there will be 0 hits for A and 0 hits for B. Therefore, in every access is a miss. That is why the miss rate will be 100%. The number of times loop is running is 32768, and the number of memory references will be 2 per loop. So, 2 x 32768 = 65536 memory reference. The entirety of the references will be misses. If we start from B [0]A[0] with cold and conflict miss, 256 times. We would have B [32760] conflict miss and A[32760] conflict miss
Miss rate = 100%
Number of Misses = 65536
Type of miss = 65280 conflict misses, and 256 cold misses.

Q5)

a) Normalize1 is faster than Normalize2 because of its data access pattern. Normalize1 accesses data in row-major order, which leads to better spatial locality and higher cache hit rates. This can be seen in the perf data, as Normalize1 has a lower number of LLC-load-misses than Normalize2, indicating that it has fewer cache misses.

b) The miss rate increases for N larger than 512 because of the increased working set size, which leads to increased cache pressure and higher cache miss rates. The intermediate effect at N=1024 can be attributed to the fact that the working set size for N=1024 is still within the capacity of the L3 cache, but there is more cache conflict over access to memory.

c) The miss rates for N=256 and N=512 are misleading because they do not reflect the data access pattern of Normalize2, which has poor spatial locality and low cache hit rates.