Homework 1

Due: Feb 3rd @ 11:59 pm (as a pdf file)

Nathan Nard (nnard2)

1. Briefly state what benefits remained to decreasing features size (i.e. making smaller transistors) when decreasing clock period (i.e. increasing clock frequency) was no longer feasible because of heat dissipation issues? In other words, why did the engineers keep decreasing feature size after 2003?

   Decreasing feature size, particularly transistor size, permitted benefits such as more transistors per die on a cpu, which roughly translates to greater processing power and provided more cost effective chip production methods. One could also argue smaller transistors results in faster "switches" as there's less distance for a signal travel across a transistor, but these "speed ups" are minimal and are deminished by long chain of gate delays. Either way though, allowing more transistors on chip means more processing cores can be placed on the chip, which permits improved computation times via parallel computing. The more cores, the more benefit parallel computing provides.

2. In your own words, explain the benefits of a cache. What are some common characteristics of programs does it take advantage of to improve memory access speeds?

   A cache provides the cpu quick (relative to main memory speed) access to recently and/or frequently used data. It's quicker by being directly attached to the CPU, being smaller in size (less travel time for signals), and taking advantage of the temporal and spatial locality of the data. Programs that reuse information or instructions frequently benefit from the use of the cache as it is much faster to read these pieces of information from the cache than reading from main memory.

3. Consider the following code:

```
for (int i = 0; i < N-1; i++)
  for (int j = 0; j < N; j++)
    A[i][j] += A[i+1][j];
```

   a. Compute the number of misses in terms of N and w (words in a cache line, where a word is 8 bytes) if the cache can only store 1 entire row of matrix A (cache size = N*sizeof(double)). You can assume that A starts at the beginning of a cache line, that A is an array of doubles, and that the cache is fully associative and is an LRU cache. How many compulsory, capacity, and conflict misses are there?

      Since 1 entire row of matrix A fits in the cache, then w = N and the miss count is: $(N^2 – N)/w = (N^2 - N)/N = (N – 1)$ misses. They are all compulsory misses, there are no capacity nor conflict misses.

b. Compute the number of misses in terms of N and w if the cache can store 2 entire rows of matrix A (cache size >= 2*N*sizeof(double)). You can make the same assumptions as part (a). How many compulsory, capacity, and conflict misses are there?

The answer is the same as part a, (N – 1) misses. They are all compulsory misses, there are no capacity nor conflict misses.

4. Consider the following code:

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

a. Will 2D-tiling (blocking) reduce the number of cache misses for the above code? Justify your answer. Assume the cache is empty, fully-associative and the arrays do not alias.
   Yes. There will be fewer misses.

b. Can you think of any optimizations (aside from tiling/blocking) that could improve the performance of this code? Rewrite the code with the optimizations. State your assumptions clearly.

   Yes, a single for loop can be used instead of two nested for loops:

```
for (int i = 0; i < N*N; i++)
    A[i/N][i%N] += B[i/N][i%N] + C[i/N][i%N];
```

   This is assuming the division and modulo computations are faster than the 2D iteration of the nested for loops.

c. What is the arithmetic intensity of this code in FLOPs/byte? You can assume all arrays store double precision data.

   2 / byte.