Homework Set 1: COL380

Kartikeya Gupta, 2013CS10231

February 3, 2016

1.
   - In an un-pipelined processor, 1000 operations are processed and for each line, a total time of $2ns + 1ns + 1ns + 1ns + 1ns + 1ns + 2ns$ is required.
     $\implies$ Total time needed = $1000 * 9ns$
     $\implies$ t = $9000ns$
   - In the given timings, the first and last stages take $2ns$ the amount of time while the other states take $1ns$ time. Because of this the time spent per instruction is $2ns$.
     The number of stages in the pipeline are 7.
     As there are 2 pipelined procesoors, the number of lines executed by each will be half of earlier = 500.
     $\implies$ Time taken for processing 500 lines by a processor = $500 * 2ns = 1000ns$.
     But we have not taken into account the time which the last instruction will spend before leaving the pipeline. This is going to be $1ns + 1ns + 1ns + 1ns + 1ns + 2ns$ = $7ns$.
     $\implies$ Total time taken = $1000ns + 7ns = 1007ns$.

     Diagram showing timeline:
     TODO: Insert table

2.
   - For the peak operation, the entire y will be present in the cache and blocks of x will keep entering and getting evicted by the cache. The present z which is being needed will also be a member of the cache.
     Lets now consider the memory access times as follows

     (a) Time needed to get the entire y in cache from DRAM:
     K amount of cache lines to be retreived = $K * 100ns$ time.
     (b) Time needed to get the entire z in cache once from DRAM:
     K amount of cache lines to be retreived = $K * 100ns$ time.
     (c) Time needed to get the entire x in cache repeatedly from DRAM:
     $16K^2/4$ number of cache lines to be retreived = $4K^2 * 100ns$

     (d) Time needed for retreiving data from the caches for the arithmetic operations:

$16K^2$ opearations taking place and for each of these 3 elements have to be accessed from the cache = $48K^2$ns

(e) Time needed for processing:
$2 * 16K^2$ operations require a time of = $32K^2 ns$.

$\implies$ Total time for this = $480K^2 + 200Kns$
Total number of instructions taking place = $32 * K^2$.
$\implies$ Number of operations per second =

$$\frac{32K^2 * 10^9}{480K^2 + 200K}$$

Approximating K to 1000, $\implies$

$$\frac{32 * 10^6 * 10^3}{480 * 10^6 + 200 * 10^3} Mflops$$

$\implies$

$$66.68 Mflops$$

- Consider the following code for multiplying the matrices

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

For multiplying 2 dense matrices in given, we have to perform $(4K)^3$ mathematical operations
When the matrices are stored in row major form, for matrix A, we need to keep a given row i of the matrix A in the cache along with the memory in C where the result is to be stored. The elements of matrix B are to be fetched column wise but if we wish to access a particular element, we will get 3 other elements which are of no use at that time. Hence for a given i and j, the entire column from matrix C needs to be accessed from the DRAM.
Let us calculate the memory access times as follows:

(a) Time spent in getting Matrix A in the cache from DRAM:
$4K^2$ number of cache lines to be retrieved = $4K^2 * 100ns$ time.

(b) Time spent in getting Matrix C in the cache from DRAM:
$4K^2$ number of cache lines to be retrieved = $4K^2 * 100ns$ time.

(c) Time spent in getting Matrix B in the cache from DRAM:
$(4K)^3$ number of cache lines to be retrieved = $64K^3 * 100ns$ time.

(d) Time needed for retreiving data from caches for arithmetic operations:
$3 * (4K)^3$ values need to be accessed from the cache = $192K^3$ ns time.

(e) Time needed for the arithmetic operations to take place:
$2 * (4K)^3$ number of arithmetic operations take place = $128K^3$ ns time.

$\implies$ Total time for this = $6720K^3 + 800K^2$ ns.
Total number of instructions taking place = $128K^3$
$\implies$ Number of operations per second=

$$\frac{128 * K^3 * 10^9}{6720 * K^3 + 800 * K^2} Flops$$

Approximating K to 1000 $\implies$

$$\frac{128 * 10^9 * 10^3}{6720 * 10^9 + 800 * 10^6} MFlops$$

$\implies$

$$19.045 Mflops$$

3.
- The value assigned to y is 1. This is because when the instruction is executed on T_0, the value of y is set to 1 and cache 2, snoops the change.
- In the case when directory based cache coherence, the value is again 1. As x is in the shared section, when it gets updated by T_0 the changes are reflected.
- There is no problem based on cache protocols and coherence in this situation. The problem here is because of a race condition.

4.
-
-
-