CS 261 Machine Organisation (Fall 2021) - Homework 10

Name 1: Alexander Tkaczyk UIN 1: 678523095

Name 2: Tameem Abdallah UIN 2: 656744567

Due Date: Wednesday Dec 1

Late Due Date: Dec 2
Total points: 100

For this homework, we will continue working with the cache simulator. You are required to prepare a short report. You can continue to work in groups of at most 2 people.

The final algorithms we will consider performs matrix multiplication, a task that crops up frequently in numerical computation. The file ***blocking.cc*** (instrumented source code is in the directory you copied in the lab) uses a clever, cache-friendly algorithm to calculate ***b*** times ***c*** and stores the result in ***a***. Compile and run this program: it displays the 10x10 product on the screen.

Now take a copy of blocking.cc and call it obvious.cc
To make the copy, type the command:

```
cp blocking.cc obvious.cc
```

Replace the central part of obvious.cc with more obvious code for matrix multiplication.
Use a single for loop to calculate each element of ***a*** as follows:

$$a_{ij} = \sum_{k=0}^{MatrixSize-1} b_{ik}c_{kj}$$

Then embed this for loop inside two outer loops which generate all combinations of *i* and *j*. Compile and run obvious.cc and check it produces the same result as blocking.cc

Now let us think about how cache-friendly these two alternative programs are. First, let us work on a larger problem by changing matrixSize to 100, blockingFactor to 20, and commenting out the lines which display the product at the end of both programs. Add instrumentation to obvious.cc, along the same lines as the instrumentation in blocking.cc.

Include the header file for compact instrumentation and recompile both programs. Run them both through the cache simulator: record the miss rates for direct mapped caches of size 4, 8, 16 and 32 KB, with block sizes of 1, 2, 4 and 8 words (*32 measurements in total*).

Finally, let us see how the computer's real, hardware cache copes with a very large matrix multiplication problem.

1. Change matrixSize to 1000 and blockingFactor to 50. Remove the instrumentation by including the header file inst none.h instead of inst compact.h.
2. Compile obvious.cc and blocking.cc, with and without optimization (with and without -O2) and record the execution time of each run (*4 measurements in total*).

To get runtimes , use command:

```
time ./a.out
```

To get more details about what the time command returns, use:

```
man time
```