KOÇ UNIVERSITY

COMP 201 - Fall 2021
Lab8 Assignment

The main aim of this exercise is to give you some insight into writing cache friendly codes. You also are going to use callgrind as a profiling tool to assess cache performance.

You are provided with a simple code that tries to take the transpose of a square matrix, and it also reports the execution time of two different methods. In the naive method, we will use a very simple matrix transposition which you must probably have seen before, however in the improved version of matrix transposition, we will use blocking technique to boost cache performance. You can find main.c in your assignment folder. It is worth having a brief look at the code before keeping on.

To begin with, let's review how we can use the valgrind tool to analyze our program. First, you need to compile the program with the debugging option enabled, something like the below code. However, in our example, Makefile is provided and you do not have to compile it manually.

```
gcc -g -ggdb main.c -o main.out
```

and now execute with valgrind as

```
valgrind --tool=callgrind --simulate-cache=yes ./main.out
```

Results will be stored on the files `callgrind.out.PID`, where PID will be the process identifier. You can read these files by using the below instruction. Do not forget to replace PID with the actual number.

```
callgrind_annotate --auto=yes callgrind.out.PID
```

It worth mentioning that although you will be provided with a lot of information by calling the above function, you can just concentrate on the cache performance and counting instruction segments in this exercise.

Please answer the following question by filling correct values in the file `result.txt` which is handed out to you with `main.c`.

Question 1:

Based on the information provided by `callgrind.out.PID`, you are able to assess the performance of the program with respect to using the cache effectively. What is the missing rate of L1 cache with respect to data (both read and write) in the whole program?

Question 2:

Please report these values in `result.txt`:

- **(a)** The Number of misses in **reading** a value in L1 cache in the naive matrixTrans and improved matrixTrans functions
- **(b)** The Number of misses in **reading** a value in L1 cache in the naive matrixTrans naive matrixTrans and improved matrixTrans improved matrixTrans functions
- **(c)** The Number of misses in **writing** a value in L2 cache in the naive matrixTrans and improved matrixTrans functions

Question 3:

Change the block according to below values in the `main.c`, and then use callgrind to report L1 missing rate with respect to data (both read and write) corresponds to each block size.

- (a) block size= 8
- (b) block size= 32

Question 4:

What would happen in cache performance if we use long as a container of our matrix variables? you do not need to use callgrind for this question.

Question 5:

Based on the information of `callgrind.out.PID` file, What does **Ir**, **Dr**, and **Dw** refer to?

Question 6:

Why does Callgrind record the count of instructions, not the actual time spent in a function?

After completing your assignment, please upload `result.txt` and the `callgrind.out.PID` corresponds to exercises 1, 2, and 3.