ECE5720 Programming Assignment 1, due February 18 Spring 2021

1. General Remarks

You are asked to write 3 standard C codes:

1. a benchmark for memory access time,
2. a simple "row-by-colum" matrix matrix multiplication,
3. a "tiled" matrix-matrix multiplication.

Your codes must compile and run on `ecelinux.ece.cornell.edu`.

- Your code must be well documented so any person with limited knowledge of C could understand what the code is doing.
- All commented out lines must provide useful informations. Remove all lines that were used for debugging or other lines not relevant to the operation of the codes.
- Codes must be saved in separate text files named `mynetid_hw1_problem_number.c`. For example `awb8_hw1_1.c` is a code written by `awb8` for Homework 1, problem 1.
- Please follow templates from the ecelinux cluster in the directory
  `/classes/ece5720/assignments/hw1`
- The first line in the text file must show your name and net id.
- You are asked to run codes for a range of parameters and measure the execution times. Results of these benchmarks need to be described in a file `mynetid_hw1.pdf` (a single file for all three codes).
- For each template there is a gnuplot (graphing software) script that generates a graph of time measurement, either as a *.pdf or *.eps file. You will incude these graphs in your Homework 1 writeup.
- Please **DO NOT** submit `*.docx` files. If submitted, those **WILL NOT** be graded.
- All files are to be archived with `tar` or `zip` programs. The archive must have the name `mynetid_hw1.suffix` where `suffix` is either `tar` or `zip`.
- Your archive **CANNOT** contain any directories or subdirectories.
- Please submit your work to Canvas.

2. Timing C codes

For timing your codes see the code templates or visit

`https://www.cs.rutgers.edu/~pxk/416/notes/c-tutorials/gettime.html`

3. Problem 1. Memory strided access time

In this assignment you will test memory access time by "touching" elements of a linear array `A` with a progressively growing stepsize (stride). You will work with arrays of a `float` type.

Define an array of length `MAX_SIZE` $= 2^{26}$ (decrease the length if `malloc` fails). Probe the access time to subarrays `A[0:n]` staring from `n = MIN_SIZE` $= 2^{10}$, then doubling `n` until `n = MAX_SIZE`. For each length, "touch" elements which are `STRIDE` $= 1$ apart, then doubling `STRIDE` until `STRIDE` becomes half of the array length. Runs for each combination of array length and stride length should be repeated 10 times and the timing for each combination should be averaged (by 10). A pseudo code might look as follows

```
for (n := 2*n)                                        (double the length of n)
  for array A[0:n-1]
    for (s := 2*s)                                    (double the stride)
      Tstart = clock_gettime()                        (start the timer)
      for (k = 1 to s*K)                              (repeat s*K times)
        for (i = 0:s:n-1)
          touch A(i)                                   (touch A[0], A[s], A[2s],..)
      T(n,s) = (clock_gettime() - Tstart)/(n*K)  (average the time)
```

Note that the loop over `k` is executed `s*K` times. This is done because we want to touch elements of `A[0:n-1]` with stride `s`, `n*K` times.

The averaged time measurements should be stored in 2D array where columns correspond to successively longer subarrays, and rows correspond to successively longer strides. For not valid combinations (like strides longer than a subarray length), record 0.00 timing. The first column is the index column (for example from `log_2(MIN_SIZE)` to `log_2(MAX_SIZE)`), which is used for graphingh as the x-axis.

All (averaged) results should be poltted on the same graph with the y-axis being the (averaged) time.

For plotting use gnuplot graphing package. Gnuplot scripts are provided. (For example, for the code `mm_tiled.c` the script could be named `plot_tiled.gp`). Open the scripts and select either the *.eps or *.pdf option for printing. The scripts contain simple explanation of commands. Feel free to adjust parameters to your taste.

As alternative to gnuplot, you may want to produce graphs in MATLAB and save them either in *.eps or *.pdf formats.

Once you have your (combined) plots ready, go to directory

```
/sys/devices/system/cpu/cpu0/cache/indexN
```

where N = 0, 2 or 3 describes data about cache level N. (index1 is for the instructions cache.) There you will be able to find

- size of cache,
- number of sets,
- associativity,
- line length,

all in bytes.

Compare your plots with cache info. Explor possible correlations between cache info and your plots. For example, do the plots correlate with

1. L1 caache size,
2. L1 cache line length,
3. L1 associativity ?

Describe what you found in your writeup.

4. Problem 2. Matrix-matrix multiplication

Recall the problem of matrix-matrix multiplication discussed in Lecture 1. There were two algorithms presented:

- an elementary "row-by-column" approach, and
- a blocked (or tiled) approach.

You are asked to write C codes implementing these two sequential algorithms. Use the templates.

4. General instructions

- there should be a single file for each code
- name the files as follows
  - `mynetid_hw1_2.c` for the row-by-column method
  - `mynetid_hw1_3.c` for the blocked method
  - `mynetid` is your Cornell net ID.
- your codes must be written in standard C language and compiled as indicated in the templates, please use only standard C directives (OS independent)
- the final version of your codes must compile and run on `ecelinux.ece.cornell.edu` servers,

- if after submission, the compiler `gcc` generates compile errors, the codes will be considered as noncompliant and will not earn any credit
- please clean the code so all spurious and debugging commands are removed
- when reporting timings as numerical values please use the scientific format `%x.xe`, where `x` is an integer.
- the memory for matrices should be allocated dynamically,
- your codes and findings (discussion of performance) must be described in a file `mynetid_hw1.pdf`, for clarity and to save space you can refer to relevant lines in the codes,
- if you relay on resources outside lecture notes but publically available, you need to cite sources in your write-up,
- submit your work on Canvas.

5. Benchmarking.

In Problem 2 you want to investigate the influence of various parameters on the performance (speed) of the codes. In your writeup, please discuss the following points.

- Mamtrix-matrix multiplication requires $O(n^3)$ floating point operations for an $n \times n$ matrices. Does the timing increses 8-fold when the matrix size is doubeld?
- How does the performance of the sequential row-by-column compare to the sequential blocked matrix-matrix multiplication?
- How does perfomance vary with different dimensions of tiles? What is the speed-up (or slow down) of the blocked version over the row by column version?
- Normalize your measurements by n^3 to see how times per a single flop change when the dimensions change.

********** **NOTE** **********

This assignment is an individual assignment. You may discuss the assignment with your classmate but the work must be your own (no code sharing). If you use outside sources for this assignment (like books, articles, internet), you must clearly cite them.