Homework #1

Submit your work work on the Blackboard before midnight the day the homework is due. Failure to do so will result in late penalties, see the syllabus for grading detail.

Here are the requirements for your Blackboard submission:

- Attach the assignment as a compressed archive file (.zip, .tgz, .tbz2, .rar)
- The name of the file should be: *firstName-lastName*-HW-*assignmentNumber.extension* (e.g. Jane-Doe-HW-1.zip)
- Include your e-mail address in the Comment field when submitting the assignment through the Digital Drop Box
- If for any reason you are submitting the assignment more than once, indicate this in the Comment field by including the word COMPLEMENT.

	Problem	Page	Points
1	See #1 below		60
2	See #2 below		110
Maximum mark (100%)			**170**

1. (a) Using trace files, i.e. files that contain addresses issued by some CPU to execute some application(s), draw the histogram of address distribution for each of them (2x20 points). On the *Ox* axis of the plot you will have the address as a decimal number. On the *Oy* axis you will have the number of occurrences for each particular address.

NOTE: the range of addresses is vast, attempting to plot everything will result in a histogram with very little detail. Instead, select a range of addresses (a few hundreds of them) where you have non-zero values on *Oy*.

Comment based on the histograms (5). The files to use are:

- cc1.din
- spice.din

The first file contains the trace obtained by the CC compiler compiling itself, and the second one comes from the running SPICE ("Simulation Program with Integrated Circuit Emphasis"), the general-purpose, open-source analog electronic circuit simulator.

Each line in the file has two fields: the first field indicates what kind of operation the CPU performs (read, write, etc.), and the second field is the address. Here is what the number in the first field means:

- 0: read data
- 1: write data
- 2: instruction fetch

The second field is the address being referenced: the address is a hexadecimal byte-address between 0 and ULONG_MAX inclusively.

(b) What is the frequency of writes (5)? What is the frequency of reads (5)? Please comment on these results (5).

HINT: Linux has lots of great utilities, such as awk, sort, gnuplot, that may come handy in solving this problem.

2. (a) Write a program, using C, C++, or Java, that multiplies two rectangular matrices -- please no square matrices -- whose elements are randomly generated. You may not use a matrix multiplication library in your code. You will have

two versions of the program, one in which matrix elements are integers and another one where they are real numbers (double) (2x15 points).

You will compile and run the programs on two different systems -- most likely one of them will be your own desktop/laptop and the other one a computer in the lab, or otherwise on one of the UNIX computers IIT makes available to its students.

Measure the time it takes each program to complete (2x5) and then compare the performance of the two systems (5). Since the matrices are randomly generated, you will have to run the program several times, measure the running time and then take the average. Also the running time has to be significantly large (at least several seconds) to reduce the impact of measuring errors; this means you will have to work with matrices that have at least hundreds of lines and columns.

Is the performance ratio the same as the clock rate ratio of the two systems (5)? Explain. Based on the retail price of the two systems, which one is more cost effective (5)?

(b) Change your multiplication algorithm and repeat the steps above; for instance, if you used the the naive multiplication algorith with the column in the inner loop, then just use the same algorithm with the row in the inner loop (same scoring as part a).

Make sure your work includes a description of the two systems (manufacturer, CPU type, amount of memory, operating system, etc.) and of the compiler used (5). Attach the source code, the tables with your time measurements for your work, and a link to your repository such that we can check-out the code, build, and execute (5).
