Preferred Networks Intern Screening 2018 Coding Task for Processor/Compiler Fields

Changelog

- 2018-5-1: Initial version

Notice

- The following programming languages must be used:
  - C or C++.
- Use the following Linux distribution, compiler and CPU:
  - Ubuntu 16.04 / gcc 5.4 / x86_64
  - Although we do not restrict the computing environment further, pick an environment and use it for all the problems.
- **(Important)** The optimization flag must be set to `-O0`. Other optimization flags must not be used.
- You are allowed to use thread parallelization, GCC extension such as inline assembly instructions, x86 built-in functions, extended CPU instructions like AVX, and so on.
  - However, you must not use external libraries except standard C/C++ libraries unless it is necessary for the above features.
- You must not use accelerators including GPU and distributed memory parallelization.
- Performance monitoring tools like perf can be used.
- Do not share or discuss any details of this coding task with anyone.
- Please tackle the task by yourself. Do not share or discuss this coding task with anyone including other applicants. If we find an evidence of leakage, the applicant will be disqualified. If one applicant allows another applicant to copy the answer, both applicants will be disqualified.
- We expect you to spend up to two days for this task. You can submit your work without solving all of the problems. Please do your best without neglecting your coursework.

Things to submit

- Submit reports for Problem 1, 2, 3 and 4.
- Source codes of each trial and Makefile in Problem 2. `make all` command to generate the execution files of the all trial.

Evaluation

It is desirable that your submission satisfies the following. (These are not mandatory. Your submission does not need to satisfy all of them if you are too busy.)

- The source code is readable for other programmers.
- There is an appropriate amount of unit tests and/or verification code to ensure the source code is correct.
- It is easy to reproduce the experimental results.
- The submitted report is concise and easy to follow.

How to submit

- Create a zip archive with all of the submission materials and submit it on this form. Due date is Monday, May 14th, 2018, 23:59 JST.

Inquiry

- If you have any question on this problem description, please contact us at intern2018@preferred.jp. An updated version will be shared with all applicants if any change is made. Note that we cannot comment on the approach or give hints to the solution.

Task description

Problem 1

Compile and run the attached file `main.cc`. Report the elapsed time shown as "average time:". Investigate the peak FLOPS (floating point operations per second) of your environment. Report execution efficiency of the program with respect to the peak FLOPS. If you find other reasons of your environment which are deeply related to the efficiency, also report them with their values.

Problem 2

Apply some steps of optimization to the function `f()` freely (you cannot change the arguments). For each step of optimization report the elapsed time and the maximum absolute error (reported as "max error:"). The error must not be larger than `1.0 x 10^-15`. The report must be organinzed in the following format for each step. The number of steps must be equal to or less than 9 (we will not evaluate the number of steps itself).

- A hypothesis on a factor that decreases execution efficiency.
- The point of a change in the source code that will resolve the factor.
- The result of the elapsed time after the change and a discussion on it.
- (Optional) a method to evaluate the impact of the factor and its result.

In addition to the trial with the best elapsed time, we will also positively evaluate a trial which makes the result worse only if the hypothesis and the discussion are appropriate. Source code must be an independent file per trial and make correspondence between file name and the trial.

Problem 3

Describe using a block diagram and procedure to implement the function `f()` in hardware.

This task is independt of the changes in Problem 2.

Problem 4 (Optional)

Design a circuit to control the hardware implementation of `f()` and describe its timing chart.

You should define the number of cycles of each arithmetic unit and memory resource appropriately.

(End of task description)