CS 4290 / CS 6290 / ECE 4100 / ECE 6100
Advanced Computer Architecture

Lab 1: Analyzing Benchmark Traces and Computing CPI (5pts)

Due: Friday, August 25th, 2023 / (11:55 pm)

This is an individual assignment. You can discuss this assignment with your classmates but you should code your assignment individually. You are NOT allowed to see the code of (or show your code to) other students. *We will use code plagiarism detection software on all submissions.*

OBJECTIVE: The objective of the first lab assignment is to test the proficiency of students in doing programming-based assignments and to ensure that the students have the basic background in computer architecture to take this graduate-level course. **The assignment is due before the Phase II registration deadline** (Friday), so that the students can make well-informed decisions whether they should continue with the course or should consider taking it after getting the required prerequisites. Given the lab's purpose, students are expected to be able to confidently complete it without any help from TAs. We will provide an autograder for this assignment, so that you can estimate the score your submission is likely to receive at the time of submission.

PROBLEM DESCRIPTION: The first lab assignment's goal is the analysis of the static and dynamic instruction occurrences in a given benchmark trace. We will provide a code template and traces from four benchmarks (gcc, mcf, libquantum, and bzip2) selected from the SPEC CPU2006 suite. We will also provide a trace reader for these traces. Your tasks are the following:

Task 1: Quantify the mix of the dynamic instruction stream. The instructions in the trace are classified as five types: ALU, LOAD, STORE, CBR (Conditional Branch), and OTHER. You will modify the provided code to count the number of dynamic instructions for each category and the percentage of these instruction types in the instruction mix.

Task 2: Estimate the overall CPI (Cycles Per Instruction) using a simple CPI model in which the CPI for each category of instructions is provided. We will use the following CPI for each category:
1) ALU: 1
2) Load: 2
3) Store: 2
4) CBR: 3
5) Other: 1

Note: As the instruction mix for each trace is likely to be different, the CPI for each trace is likely to be also different.

Task 3: Estimate the instruction footprint by counting the number of unique PCs in the benchmark trace. Normally, this information should be multiplied by the average bytes per instruction to get the total footprint, however for this lab we will forego this multiplication.

HOW TO GET STARTED:

1. Download the tarball (Lab1.tar.gz) from Canvas.
2. "tar -xvzf Lab1.tar.gz"
3. "cd Lab1"
4. "ls" -- Lab1 contains four subdirectories: src, scripts, traces, and results.
5. "cd src"
6. "ls" (there are three files: makefile, trace.h, sim.cpp)
7. Open sim.cpp in your favorite editor. We have already provided the trace reader and print functions. Your objective is to simply write the function analyze_trace_record() and update the stats variable (stat_*).
8. Once you write the function, type "make" … this should create an executable "sim".
9. "./sim ../traces/bzip2.otr.gz" (to run one trace and see the output).
10. Perform a sanity check on the output to verify whether it makes sense or you need to debug your code.
11. Once your code is ready, go to the scripts directory "cd ../scripts".
12. "./runall.sh" -- this runs your code for all four traces, stores the result files in the results directory and also generates the "Report.txt" file.
13. The last line of the report file is your approximate score out of 5 points. Note that we will run your source code on a separate set of "hidden" traces, so your graded score may be different if your implementation is incorrect.

WHAT TO SUBMIT (on Gradescope): A single file ONLY: Your **sim.cpp.**

REFERENCE MACHINE:
We will use the virtual machine **oortcloud.cc.gatech.edu** as a test machine for this course. You should be able to connect to it using ssh and transfer files between oortcloud and your local machine using scp. If you are not officially enrolled in the course yet, you may not be able to access the machine. In that case, make sure that your code works on a standard Linux machine. We will be compiling and running your lab submissions using an autograder on Gradescope (based on this docker image). We recommend using oortcloud for testing/debugging and submitting to gradescope for checking the points.

Warning: Before submitting your code on Gradescope ensure that your code compiles on this machine and generates the desired output (without any extra printf statements). Please follow the submission instructions. If you do not follow the submission file names, you will not receive full credit.

NOTE 1: It is impractical for us to support other platforms such as Mac, Windows, Ubuntu etc.

HOW WELL DID YOU DO? If you have the right background, this lab should take about 1-2 hours to complete. If this lab assignment takes you 5-10 hours or more, you may not have the right background for this course. Labs 2, 3 and 4 will each likely take about 10x more time and are about 10x harder. So, if you do not have the right background, you may find the assignments for this course to be extraordinarily difficult.

General advice for all labs:
We *strongly* recommend that you use a *private* git repository to version-control your code and prevent any loss. You should never assume that any code/data stored on a local machine (your laptop, desktop, or even the oortcloud machine) are safe. Although data loss is very rare, you should always follow best practices to completely eliminate that chance. We will not accept incidents like "my laptop broke down/oortcloud was wiped and I lost all my progress" as a valid excuse for late homework submission.

FAQ:

1. **/usr/bin/ld: cannot find -lz collect2: error: ld returned 1 exit status**
   Please remove the '-lz' flag from the Makefile or install zlibc on your system.

2. **How do I connect to the reference machine?**
   a. You will need to connect to VPN using a VPN client to access most machines at GT: https://faq.oit.gatech.edu/content/how-do-i-get-started-campus-vpn
   b. An ssh client (e.g., Putty SSH) to connect from your machine to the reference machine.
   c. SCP for file transfers to/from the server. There are several scp GUI tools available or you can directly execute the command from the terminal (https://linux.die.net/man/1/scp).

3. **Why can't I execute the runall.sh script?**
   Check that you have execute permissions on the file. Add execute permissions to runall.sh and genreport.oortcloud using chmod +x.