Due: Friday, August 26, 2022, 11:59pm ET

ECE 4100/ECE 6100/CS 6290
Advanced Computer Architecture
Lab 1: Analyzing Benchmark Traces and Computing CPI (5pts)
Due: Friday, August 26, 2022, 11:59pm ET

This is an individual assignment. You can discuss this assignment with other classmates but you should code your assignment individually. You are NOT allowed to see the code of (or show your code to) other students.

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 the day of 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. We will provide an autograder (**runall.sh**) for this assignment, so that you can estimate the score your submission is likely to receive at the time of submitting.

PROBLEM DESCRIPTION: The first lab assignment is aimed at doing analysis of the static and dynamic occurrences of instructions in a given benchmark trace. The benchmark traces are a capture of what instructions the CPU performed during a single run of a given program.

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 job is to do 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 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 different also.

Task 3: Estimate the instruction footprint by counting the number of unique PCs in the benchmark trace (ideally, 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:
This lab is designed to be run on the reference machine. (See the "Reference Machine" section below to identify yours.)

1. Extract the tarball: `tar -xvzf Lab_1.tar.gz`
2. Change into the extracted directory: `cd Lab_1`
3. `ls` — Lab_1 contains four subdirectories: src, scripts, traces, and results
4. `cd src`
5. `ls` (there are four files: Makefile, trace.h, sim.cpp, studentwork.cpp)
6. Open studentwork.cpp in your favorite editor (e.g. `nano studentwork.cpp`) – this is where you'll write the function `analyze_trace_record()` to update the stats variables (`stat_*`).
    a. The functions to read the trace files and print the final stats are already provided for you in sim.cpp. Do not modify this file.
    b. Types are defined in trace.h. Do not modify this file.
7. Once you write the function, type `make` — this should create an executable `sim`
8. `./sim ../traces/bzip2.otr.gz` (to run one trace and see the output)
9. Do a sanity check to see if the output makes sense (or if you need to debug your code)
10. Once your code is ready, go to the scripts directory: `cd ../scripts`
11. Do `./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. (This may take a minute!)
12. 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 not correct.

WHAT TO SUBMIT: Two files ONLY: Your **studentwork.cpp** and **report.txt**.
Do NOT include sim.cpp or trace.h. Your code in studentwork.cpp MUST work with the unmodified versions of these files.

NOTE: If you do not have access to Canvas before the submission deadline, email the two files **studentwork.cpp** and **report.txt** to nwong36@gatech.edu. Include your GT username in the email.

REFERENCE MACHINE:
STUDENTS IN ECE 4100/6100 SECTION: We will use **ece-linlabsrv01.ece.gatech.edu** as the reference machine for this course. (https://help.ece.gatech.edu/labs/names)

Undergraduate ECE students, non-ECE students, or those who otherwise do not have access may need to request activation of their ECE Linux account to connect to this server; see FAQ #2 below.

Due: Friday, August 26, 2022, 11:59pm ET

STUDENTS IN CS 6290 SECTION: We will use **shuttle3.cc.gatech.edu** as the reference machine for this course. (https://support.cc.gatech.edu/facilities/general-access-servers)

Before submitting your code ensure that your code compiles on the reference machine for your section 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 the full credit.
(Note that Canvas appends a number to file names whenever you resubmit, e.g. report-1.txt and studentwork-1.cpp. This is acceptable and will not be penalized.)

NOTE: 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 it takes you 5–10 hours or more to do this lab assignment, 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.

FAQ:

1. **How do I connect to ece-linlabsrv01.ece.gatech.edu or shuttle3.cc.gatech.edu?**
    a. You may need to install a Georgia Tech VPN client (for access to any machine at Georgia Tech).
        i. Please be aware that accessing ece-linlabsrv01.ece.gatech.edu typically requires the use of the VPN, *even when on the campus network.*
    b. On Windows, you can use the PuTTY SSH client to ssh from your machine to the server. (A command-line ssh tool is typically installed by default on other platforms.)
    c. Use 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.

2. **I get "Permission denied" or "Access denied" when trying to connect to ece-linlabsrv01.ece.gatech.edu.**
    If you are unable to access this machine with your GT username and password, you may not have an ECE Linux account. Contact help@ece.gatech.edu specifying that you need to activate your ECE Linux account and including your GT username, the course for which the account is required (ECE 4100 or 6100), and the instructor for the course (Prof. Callie Hao).

3. **I get an error when I try to execute the runall.sh script.**
    Check that you have execute permissions on both runall.sh and genreport.ecelinsrv7. Add execute permissions to these files using the command:
    `chmod +x runall.sh genreport.ecelinsrv7`

4. **I get an error "`unrecognized command-line option "-std=c++11"`" when compiling on shuttle3.cc.gatech.edu.**
   In the Makefile, change the line:
   `CXXFLAGS=-g -std=c++11 -Wall`
   to the following:
   `CXXFLAGS=-g -Wall`
   This will allow your code to compile if you avoid using any C++11 features.
   Don't worry about submitting the modified Makefile. If needed, we will apply this modification when grading without any additional action on your part.