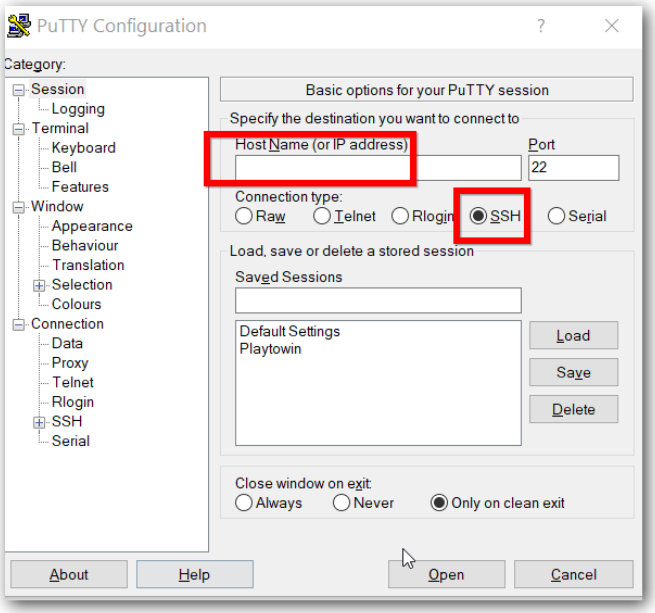
Name: _____

Lab for ITSC 3181 Introduction to Computer Architecture, Spring 2023

Lab #03: Remote access to Linux machine, Basic C programming and performance with sum. The purposes of this lab are 1) to get comfortable with C compilation steps, 2) to use perf command for collecting metrics of program execution, and 3) to use the CPU ExeTime formula to model program execution time based on the perf output. Due on 02/01. Total points: 1% of the final accumulated percentage grade.

Before the lab, make sure you are able to connect to the university network via VPN and are able to login to an educational cluster named Centaurus (https://oneit.charlotte.edu/urc/educational-clusters) available as hpc-student.uncc.edu. Follow the following steps to do it:

1. If UNCC VPN has not been setup on your computer, following these steps to connect to UNCC network https://spaces.uncc.edu/pages/viewpage.action?pageId=6653379
2. To login one of the two machines, you will need a tool called putty (or others) if you are using Windows OS. Download putty from https://www.putty.org/ , which is a putty.exe executable and launch the executable.
1. Then from the putty window, see below, input "hpc-student.uncc.eduin the "Host Name" box and make sure SSH is selected as Connection type. The program will prompt you a login window from which you input your UNCC credential (id and password). You need to have your DUE setup in order to be authenticated. Check https://spaces.uncc.edu/pages/viewpage.action?pageId=35651686.



2. If you are using Mac OS X, open the Mac OS X terminal app from Finder→Application→Utilities and type in command "ssh -l<YourUNCCID> hpc-student.uncc.edu". For example, my UNCC id is yyan7, I type in "ssh -lyyan7 hpc-student.uncc.edu. Then you will be prompted with password. After you key in your password and hit return, you should be in the terminal of the computer with a flashing cursor that you can type in Linux command. You need to have your DUE setup in order to be authenticated. Check https://spaces.uncc.edu/pages/viewpage.action?pageId=35651686.
3. To report connection issue, please use https://servicecatalog.charlotte.edu/service/research-computing/high-performance-computing-research , you should specify in details (your UNCC ID), whether your issue is from VPN/credential, or the ssh issue to access the educational cluster for ITSC 3181.

During the lab:

1. Follow the steps above to login in hpc-student.uncc.edu.
2. Check source code and assembly code from https://passlab.github.io/ITSC3181/exercises/sum
3. On your Linux login terminal, type in sum_full.c using vim (or other editor such as nano) and understand the source code while typing. https://passlab.github.io/ITSC3181/exercises/sum/sum_full.c. Understand how to use the read_timer function to measure the execution time of a specific portion of your program.
4. Compile the sum_full.c file step-by-step using -E, -S, -c option and then finally link to create executable. In each step, the compiler performs the corresponding compilation operation which can be shown by using the "gcc --help" command. You can check for more details about those steps from slides for Compilation, Assembling, Linking and Program Execution (https://passlab.github.io/ITSC3181/notes/lecture06_CompilationAssemblingLinkingProgramExecution.pdf). Check the intermediate compilation output and do not move on the next step unless previous step is successful.

```
-E                      Preprocess only; do not compile, assemble or link
-S                      Compile only; do not assemble or link
-c                      Compile and assemble, but do not link
-o <file>               Place the output into <file>
```

   vi sum_full.c
   gcc -E sum_full.c -o sum_full.i
   gcc -S sum_full.i
   gcc -c sum_full.s
   gcc sum_full.o -o sum

   "gcc -save-temps sum_full.c -o sum" can do all the steps at once and also produce all the intermediate files. But in this exercise, you should use the flag to compile step by step.

5. Execute the sum and time it:

   ./sum 100000
   time ./sum 10000000

6. Execute the sum program with perf command. (Check http://www.brendangregg.com/perf.html for more perf command)

   perf stat ./sum 100000

7. Use perf command to collect execution profile of sum_full.c program. After collecting execution profile, which includes number of cycles, number of instructions and CPU frequency, calculate the number of cycles and instructions for the loop of the sum method itself. Then calculate the CPI and CPU time (ms) using the formula we learned during the class (check slides from CPU Performance and Power (https://passlab.github.io/ITSC3181/notes/Chapter01_AbstractionTech_PerformancePower.pdf) or textbook 1.6). Fill in the following table using the information you collect or calculate. The CPU time (ms) outputted by the program itself is the Runtime (ms) column of the program output. For the following table columns, The CPU Time you calculate is the modeled execution time and the CPU Time output from the program is the actual measured time. By comparing these two columns, you should be ablet o see the difference between modeled time and measured time. Important, for perf, cycles and instructions collected from perf.

Submission: Submit your work using the submission page that you can find from https://passlab.github.io/ITSC3181/notes/Lab_03_PerfSum_SubmissionPage.docx . The submission page has the following table. Your submission should have all the cells filled.

Size	#Cycles	# Instructions	CPI	CPU Freq (Clock Rate) (GHz)	CPU Time (ms) for sum loop (#cycles/Clock Rate and convert it to ms)	CPU time (ms) outputted by the program itself
0						
100						
1000						
10000						
100000						
1000000						
10000000						