Lab 3 Specification – Exploring Additional Performance Assessment Techniques
Due (via your git repo) no later than 2 p.m., Wednesday, 30th September 2020.
50 points

Lab Goals

- Practice a few more performance assessment problems.
- Reflection on the architecture used in computing.
- More practice with C programming.

Summary

You will do a few exercises just to refresh your understanding and practice Performance Assessment techniques discussed in class, and extend the learning from in-class activities. We will watch a video clip, where the architecture of computing is discussed. Von Neumann Architecture is how nearly all computers are built, but who was John Von Neumann and where did the architecture come from? is discussed in the video. We will take up an additional programming exercise to further expand our knowledge in C programming (in an un-explored low-end territory).

Learning Assignment

If you have not done so already, please read all of the relevant ”GitHub Guides”, available at https://guides.github.com/, which explains how to use many of the features that GitHub provides. In particular, please make sure that you have read guides such as ”Mastering Markdown” and ”Documenting Your Projects on GitHub”; each of them will help you to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, you should also read

- PH chapter 01 - section 1.3, 1.5, 1.6 - 1.7, 1.10
- KR chapter 01 - section 1.1 - 1.5
- Class discussion notes, slides, and in-class coding files.

Assignment Details

Now that we have discussed some fundamental principles behind performance assessment at the hardware-level, and got familiarized with the C programming language, it is now time to implement some challenging requirements of performance assessment. In this process, we will also implement a tool using a C program to evaluate the performance based on user-provided metrics. Additionally, we will watch and reflect on a video to learn some intricacies of the C programming language, to retain the knowledge from the class discussions so far.

It is required for all students to follow the honor code. Some important points from the class honor code are outlined below for your reference:

1. Students are not allowed to share code files and/or other implementation details. It is acceptable to have a healthy discussion with your peers. However, this discussion should be limited to sharing ideas only.
2. Submitting a copy of the other's program(s) is strictly not allowed. Please note that all work done during laboratory sessions will be an opportunity for students to learn, practice, and master the materials taught in this course. By doing the work individually, students maximize the learning and increase the chances to do well in other assessments such as lab assignments, skill tests, projects, etc.

At any duration during and/or after the lab session, students are recommended to team up with the Professor and/or the Technical Leader(s) to clarify if there is any confusion related to the items in the lab sheet and/or class materials.

Section 1: Performance Exercise

This section is worth 20 points. The points breakdown is provided below:

- Task 1 = 10 points
- Task 2 = 10 points

Performance assessment is an important technique that provides the right set of tools to compare machines at the hardware level and guide us in the process of identifying the performance of computers. This procedure is critical in comparing two or more machines and identifying the fastest or slowest machine at the hardware level. One of the recent additions to our discussion on performance assessment, is the Amdhals Law.

In computer architecture, Amdahl's law (or Amdahl's argument[1]) is a formula that gives the theoretical speedup in latency of the execution of a task at a fixed workload that can be expected of a system whose resources are improved. It is named after computer scientist Gene Amdahl, and was presented at the AFIPS Spring Joint Computer Conference in 1967.

Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. For example, if a program needs 20 hours to complete using a single thread, but a one-hour portion of the program cannot be parallelized, therefore only the remaining 19 hours ($p = 0.95$) of execution time can be parallelized, then regardless of how many threads are devoted to a parallelized execution of this program, the minimum execution time cannot be less than one hour. (text is taken from **wiki**).

To practice and retain knowledge from class discussions, it is required to complete the following tasks, outlined below:

- **Task 1:** From the 5th edition of the Patterson and Hennessy textbook, please answer all parts to Exercise 1.13 on page 58. It is expected that you present your solution as detailed as possible. You can consider, the level of details to be similar to the examples provided in slides. Prepare your solution file using a markdown file and save the file as `sol-1` in your repository.

- **Task 2:** From the 5th edition of the Patterson and Hennessy textbook, please answer all parts to Exercise 1.14 on page 59. It is expected that you present your solution as detailed as possible. You can consider, the level of details to be similar to the examples provided in slides. Prepare your solution file using a markdown file and save the file as `sol-2` in your repository.

Section 2: Computing Architecture

This section is worth 10 points. The points breakdown is provided below:

- Task 3 = 10 points, a maximum of 2 points awarded for each question.

Von Neumann Architecture is how nearly all computers are built, but who was John Von Neumann and where did the architecture come from? is discussed in the video. To complete this part, it is required to do the following:

- **Task 3:** Watch this 10-minute video, by using the link below:

  `https://www.youtube.com/watch?v=Ml3-kVYLNr8&t=345s`

  After watching the video, create a markdown file and name it as `video-reflection`. In this file, provide a detailed description of the questions presented below:

  1. Who is Von Neumann? and how did he contribute to modern computing?
  2. Explain the different components of Von Neumann architecture, and how do they work together based on the discussion points mentioned in the video?
  3. What did the speaker say about storing instructions and data in memory?
  4. What is ENIAC? and how is Johnny related to this project?
  5. What is EDVAC? and how is this different from ENIAC?

Section 3: Experiencing an unexplored territory of C

This section is worth 20 points. The points breakdown is provided below:

- Task 4 = 10 points
- Task 5 = 10 points

It is worth making a note, that the file system is just the output component referenced in the Von Neumann architecture. So far, we primarily used the Display monitor screen, as the output component to process our programs. In this section, we will experience an unexplored territory of developing C programs using the File System. To complete this part, it is required to have a solid understanding of the basic concepts of C Programming from class discussions and the previous two labs. It is important to review the slides, reason through and understand the logical tasks (binary using modulo 2) discussed, and also complete the reading assignments as required. There is a C program provided in the starter-code provided repository, to help stay focused on the implementation details, and to get started with the development of this part. To complete this part, it is required to do the following:

- On page 17 of K&R there is a program to copy a file. To run it on a file named “myfile.txt”, you type:

  ```
  ./a.out < myfile.txt
  ```

  Review the starter-code in a file named `parsefile.c` and `ritchie.txt`. The code is modified in the `parsefile` program so that it prints out only every other character (other than the newline character, which will always be printed), i.e., the first, third, fifth, seventh,$\cdots$ This implementation is done by adding a counter and only printing when the counter has an even value. The text file named ”ritchie.txt” can be used for test data, which is also available in the ”resources” directory. Read the comments to make sure you understand how the program works. prompts to get the user input. This understanding is important to complete the tasks outlined below:

  1. **Task 4:** Write a C program modeled after the ones in sections 1.5.3 and 1.5.4 of K&R that do the following: Given an input file, print it with leading line numbers, starting with line 1. Assume there are no more than 999 lines in the file. Line numbers should be right-justified in the first 3 columns. Submit your work, using a new file named `file1.c`.
  2. **Task 5:** Write a C program modeled after the ones in sections 1.5.3 and 1.5.4 that does the following: Given an input file, count the number of vowels and consonants and print out these counts, appropriately labeled. The vowels are a, e, i, o, and u (both upper and lower case); all other letters are consonants. You may not use any built-in C functions for checking for upper-case, etc. Submit your work, using a new file named `file2.c`.

     The text file named ”ritchie.txt” can be used for test data, which is also available in the ”resources” directory.

Submission Details

For this assignment, please submit the following to your GitHub lab repository.

1. **video-reflection** markdown file
2. upload of **sol-1** file
3. upload of **sol-2** file
4. updated version of **file1.c** file
5. updated version of **file2.c** file
6. It is highly important, for you to meet the honor code standards provided by the college and to ensure that the submission is made before the deadline. The honor code policy can be accessed through the course syllabus. Make sure to add the statement ”This work is mine unless otherwise cited.” in all your deliverables such as source code and PDF files.
7. It is recommended to upload a summary file, with the details that you would like the Professor to know while grading the work. For example, it may be a reflection of your experience in the lab by highlighting some of the challenges that you had faced and a brief mention of how you addressed those challenges while implementing this lab. The summary file may also include a brief mention of any details that one should know about executing your program and what to expect during the execution.

Grading Rubric

1. Details including the points breakdown are provided in the individual sections above.
2. If a student needs any clarification on their lab credits, it is strongly recommended to talk to the Professor. The lab credits may be changed if deemed appropriate.