Lab 06 Specification – Performance Assessment, and Data Translation.
Due (via your git repo) no later than 2 PM, Friday, 8th October 2021.
50 points

Lab Goals

- Exploring performance assessment strategies.
- Implementing performance metrics and data translation algorithms using C.

Learning Assignment

If not done previously, it is strongly recommended to read all of the relevant "GitHub Guides", available at the following website:
`https://guides.github.com/`
that explains how to use many of the features that GitHub provides. This reading assignment is useful to understand how to use both GitHub and GitHub Classroom. To do well on this assignment, it is also recommended to do the reading assignment from the section of the course textbook outlined below:

- **KR chapter 5 - [5.6 - 5.9] and chapter 06 - [6.1 - 6.4], PH Chapter 1.6, 1.7**

Assignment Details

In this lab, we will also have an opportunity to do more practice using performance assessment strategies and data translation techniques.

At any duration during and/or after the lab, students are recommended to team up with the Professor and the TL(s) to clarify if there is any confusion related to the lab and/or class materials. The Professor proofread the document more than once, if there is an error in the document, it will be much appreciated if you can communicate that to the Professor. The class will be then informed as soon as possible regarding the error in the document. Additionally, it is highly recommended that students will reach out to the Professor in advance of the lab submission with any questions. Waiting till the last minute will minimize the student's chances to get proper assistance from the Professor and the Technical Leader(s).

Students are recommended to get started with this part in the laboratory session, by discussing ideas and clarifying with the Professor and the Technical Leader(s). It is acceptable to discuss high-level ideas with your peers, while all the work should be done individually. Late submission is accepted for the part(s) in this section, based on the late policy outlined in the course syllabus.

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) and technical reports is strictly not allowed. Please note that all work done during lab 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 Quizzes, exams, etc · · ·

Preliminary Steps

It is important that you can set up Docker and GitHub to complete the rest of the lab. Please follow the guidelines below to complete the preliminary steps.

1. **[Docker Setup.]** At this point, I expect the MAC, Linux, and Windows Pro users, to have this step completed based on our previous class discussions. For those who had not completed this step, the documentation below should provide more details regarding the download and installation setup.

   - Get Docker setup completed on your laptops:
   - Docker Mac Setup:

     `https:/docs.docker.com/docker-for-mac/install/`
   - Docker Ubuntu Setup

     `https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04`
   - Docker Windows Setup:

     `https:/docs.docker.com/docker-for-windows/install/`
   - If the setup goes correctly as desired, you should be able to get started and validate the Docker version and run the hello world docker container using the following commands:

     docker –version

     docker run hello-world
   - There are some more documentation for Docker get started to test your installation in the link provided below:

     `https:/docs.docker.com/docker-for-mac/`

     `https:/docs.docker.com/docker-for-windows/`

2. **[Loading Docker Container.]** There are two steps in loading the container, namely:

   - Build the container
   - Connect and Run the container

   Build the container: So to build the container. the following steps should be performed.

   (a) First, accept the lab URL provided in Slack. After downloading the lab folder from the GitHub classroom, navigate to the cmpsc200-fall-21-lab06 directory using terminal (Mac/Ubuntu) or Command Prompt/Docker quick start terminal (windows).

   (b) Build the docker image using the following command:

   docker build -t cs200lab06 .

   Please note, you are required to have the period in the command above.

(c) Note: In the command above, cs200lab06 is the user-provided image name. This could be random. But it is recommended to use the same name to easily follow the rest of this document. Additionally, it is required to be inside the `cmpsc200-fall-21-lab06` directory to run the build command. If you are not inside the `cmpsc200-fall-21-lab06` directory, you may receive an error message.

(d) Upon successful build, it is recommended to verify the correctness of image creation by using the following command:

docker image ls

(e) The image named "cs200lab06" should be listed as one of the outputs from the command above.

Connect and Run the container: So to create and run the container. the following steps should be performed.

(a) Run the docker container based on the image created in the previous steps using the following command:

Mac/Ubuntu:

```
docker run --rm -v $(pwd)/src:/root -it cs200lab06
```

Windows:

```
docker run --rm -v "%cd%/src":/root -it cs200lab06
```

(b) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab06` directory. And, please note, you will log in to the container after entering the above command.

(c) After creating the container, the run command above creates a mount between the host machine and the container with a shared folder space. So, any files placed inside the host mount directory can be easily accessible inside the container mount directory and vice versa.

(d) After connecting to the container, we can compile C Programs using the command below within the container:

gcc hello.c -o hello.out

(e) After compiler the program, we can execute C Programs using the command within the container:

./hello.out

(f) **[GitHub Setup.]** Take a look at the detailed documentation for getting started with GitHub, which is available at: `https://docs.github.com/en/get-started`

You are required to know the procedure to git clone, git pull, git add, git commit, and git push to access the lab specification folder and to submit your lab for grading purposes. If there is an issue with your GitHub setup please discuss it with your Technical Leader(s) and/or the Professor.

Section 1: Performance Assessment

This section is worth 20 points.

We discussed Performance Assessment strategies, during our last few classes. Please refer to the lesson5 part-1 and part-2 slides and activities. Complete the task by adding your detailed answers to the `Technical-Report` file.

Task 1: Consider three different processors P1, P2, and P3 executing the same instruction set. P1 has a 3 GHz clock rate and a CPI of 1.5. P2 has a 2.5 GHz clock rate and a CPI of 1.0. P3 has a 4.0 GHz clock rate and has a CPI of 2.2.

1. Which processor has the highest performance expressed in instructions per second?
2. If the processors each execute a program in 10 seconds, find the number of cycles and the number of instructions.
3. We are trying to reduce the execution time by 30% but this leads to an increase of 20% in the CPI. What clock rate should we have to get this time reduction

Task 2: Compilers can have a profound impact on the performance of an application. Assume that for a program, compiler A results in a dynamic instruction count of 1.0×10^9 and has an execution time of 1.1 s, while compiler B results in a dynamic instruction count of 1.2×10^9 and an execution time of 1.5 s. Find the average CPI for each program given that the processor has a clock cycle time of 1 ns.

1. Assume the compiled programs run on two different processors. If the execution times on the two processors are the same, how much faster is the clock of the processor running compiler A's code versus the clock of the processor running compiler B's code?
2. A new compiler is developed that uses only 6.0×10^8 instructions and has an average CPI of 1.1. What is the speedup of using this new compiler versus using compiler A or B on the original processor?

Section 2: Data Translator

This section is worth 20 points.

Data conversion from decimal to binary (and vice-versa) is done by our computers, every nanosecond, tirelessly. In this part, we will implement two distinct methods in the Data Converter tool. These methods are expected to perform decimal to binary and binary to decimal conversions. 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 labs.

It is not allowed to use any external built in functions to do this translation and you are **required** to implement the algorithm. Although not required you may use one or more Static or Dynamic arrays for storage and processing purposes. This implementation could be done without using Arrays and Pointers. It is important to review the slides, reason through and understand the logical outflow of the algorithms discussed in class, and also complete the reading assignments as required. There is a C program provided in the starter-code 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:

1. Review the starter-code in a file named `translator.c`. Execute the code a few times to understand the program flow in the code file. The overall goal of this program is to implement a data conversion tool that converts a decimal to binary (and vice-versa). Read the program completely, 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:

2. **Task 3:** Complete the **b2d** method for binary to decimal translation by implementing the following:

   - Introduce a new variable called temp (of integer type) and assign the value of binary to temp.
   - Iterate, with the use of a while loop, to compute the number of bits in the given input (that is in temp). This may involve dividing temp over 10 repeatedly.
   - Iterate, with the use of another new while loop, to implement the binary to a decimal algorithm. It is worth noting the highlight of this algorithm is that we need to get each bit separately, and multiply it by 2, and add to the result repeatedly. This process will end once we reach the last bit and we terminate the while loop. To implement this part, you may need to develop an if-else condition to implement the corresponding logic inside the while loop and insert the right values in the variable named `res`. By doing the above steps, this method should return the final output in a decimal format using the variable `res`.

3. **Task 4:** Complete the **d2b** method for decimal to binary translation by implementing the following:

   - Iterate, with the use of a while loop, to compute the binary equivalent for the given decimal. You may need to repeatedly divide `decimal` over 2, till we cannot divide anymore. We may use a similar approach as we did to build the phone numbers in the past labs (digit by digit) conversion to get the output. That is simply orchestrate the output by combining all the remainders together.
   - To implement this part, you may need to develop an if-else condition to implement the corresponding logic inside the while loop and insert the right values in the variable named `res`. Hint: We need to use the modulo and pow operators to solve this part. The if-else condition may be used as an indicative factor to break out of the loop.

Section 03: Data Conversions

This section is worth 10 points.

Data conversion is the conversion of data by computers at different levels of granularity. For example: data is displayed and represented in different ways, namely:

1. decimal for human users to read, and write (in code).

2. binary for machines to read and write.

3. hexa-decimal for managing memory, using a wider range of memory addresses.

In this section, we will practice the data conversion schemes using the algorithms discussed in lesson-6 slides. Add your detailed answers to the `Technical Report` file. To practice and retain knowledge from class discussions, it is required to complete the following tasks, outlined below:

- **Task 5:** Solve the following translation problems:
  1. Convert $(11101000)_2$ to Decimal
  2. Convert $(298)_{10}$ to Binary
  3. Convert $(BEAF)_{16}$ to Decimal
  4. Convert $(987654321)_{10}$ to HexaDecimal
  5. Convert $(DECADE)_{16}$ to Binary
  6. Convert $(10001111010111101011101)_2$ to HexaDecimal

Section 04 - Honor Code

Make sure to **Sign** the following statement in the `honor-code.txt` file in your repository. To sign your name, simply replace Student Name with your name. The lab work will not be graded unless the honor code file is signed by you.

This work is mine unless otherwise cited - Student Name

Section 05 - Reflection

Add a Reflection to the repository by modifying the `reflection` file in the lab repository. List out the biggest learning points and any challenges that you have encountered during this lab.

Submission Details

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

1. updated version of **translator.c** file.
2. A document containing the technical points in the file named `Technical-report`.
3. A document containing the reflection of the lab in the file named `Reflection`.
4. A signed honor code file, named `Honorcode`.
5. To reiterate, it is highly important, for you to meet the honor code standards provided by the college. The honor code policy can be accessed through the course syllabus.

Grading Rubric

1. There will be full points awarded for the lab if all the requirements in the lab specification are correctly implemented. Partial credits may be awarded if deemed appropriate.
2. Failure to upload the lab assignment code to your GitHub repository will lead to receiving no points given for the lab submission. In this case, there is no solid base to grade the work.

3. There will be no partial credit awarded if your code doesn't compile correctly. It is highly recommended to validate if the correct version of the code is being submitted before the due date and make sure to follow the honor code policy described in the syllabus. If it is a late submission, then it is the student's responsibility to let the professor know about it after the final submission in GitHub. In this way, an updated version of the student's submission will be used for grading. If the student did not communicate about the late submission, then automatically, the most updated version before the submission deadline will be used for grading purposes. If the student had not submitted any code, then, in this case, there are no points awarded to the student.

4. If a student needs any clarification on their lab grade, it is strongly recommended to talk to the Professor. The lab grade may be changed if deemed appropriate.