Lab 09 Specification – Logical Operators and MIPS starter
Due (via your git repo) no later than 2 PM, Friday, 12th Nov 2021.
50 points

Lab Goals

- Implement logical operators such as 5 bit adder and subtractor.
- Solidify our understanding of Logic operators.
- Getting started in Assembly Language Programming (MIPS).

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

- Section 3.3 in **PH**
- Principles of computer hardware by Alan Clements - Chapter 02 - 2.6;
- Computer Organization and Design by Patterson and Hennesssy - Appendices Section B - B.3;
- Class discussion notes, slides, and in-class coding files.

Assignment Details

Now that we have discussed some fundamental principles behind low-end operators such as the logical add and subtract, and got a lead towards the development of logical operators, it is now time to implement some challenging requirements from an operational perspective. In this process, we will also take a lead to assembly language programming, also, known as MIPS.

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-lab09 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 cs200lab09 .

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

(c) Note: In the command above, cs200lab09 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-lab09` directory to run the build command. If you are not inside the `cmpsc200-fall-21-lab09` 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 "cs200lab09" 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.
Connect Run new container: So to create and run the container. the following steps should be performed.

(a) If you have a windows laptop, you can run the following command from the command prompt. Please note only the build command need to be run using the Powershell and the run container command need to be run using the command prompt. If you have a Mac or Ubuntu you can both build and run the container using your terminal window.

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

Mac/Ubuntu:

```
docker run -t -d -P -v $(pwd)/src:/root/src --rm -ti -p 5900:5900
--name lab09 cs200lab09
```

Windows:

```
docker run -t -d -P -v "%cd%/src":/root/src --rm -ti -p 5900:5900
--name lab09 cs200lab09
```

If you get an error while running these commands, then the port may be in use. You may want to try to change the port from 5900 to 5910. [Only on the left side]. For example:

Mac/Ubuntu:

```
docker run -t -d -P -v $(pwd)/src:/root/src --rm -ti -p 5910:5900
--name lab09 cs200lab09
```

Windows:

```
docker run -t -d -P -v "%cd%/src":/root/src --rm -ti -p 5910:5900
--name lab09 cs200lab09
```

Keep in mind that if you change it to 5910 then you need to use that port when connecting using VNC viewer.

(c) To run the above command, it is required to be inside the `cmpsc200-fall-21-lab09` directory. And, please note, you will log in to the container after entering the above command.

(d) 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.

(e) **[VNC Viewer Setup.]** We will use VNC Viewer to access the container. This lab requires the use of Graphical User Interface (GUI). Unlike the previous labs, we need to connect differently to access the GUI applications inside the docker container. You are required to download and install VNC Viewer. Take a look at the detailed documentation for downloading and installing VNC Viewer at:

Mac:

`https:/www.realvnc.com/en/connect/download/viewer/macos//`

Windows:

`https:/www.realvnc.com/en/connect/download/viewer/windows//`

Ubuntu:

`https:/www.realvnc.com/en/connect/download/viewer/linux//`

(f) After successful installation, open the VNC viewer and create a new connection.

(g) In the Server section, type in the following if you are a MAC or Linux user:

localhost:5900

(h) At this point, this should take you to a prompt for providing the password. The password is **1234**.

(i) After providing the password, this should take you to the container console. Congratulations, you are physically connected to the Docker container and we should be able to see the GUI at this point.

(j) At this point, you can run the following command in the console to open the logisim tool:

java -jar logisim.app/Contents/Resources/Java/logisim.jar

Section 1: Logical Operators

This section is worth 40 points. The points breakdown is provided below:

- Task 1 = 10 points
- Task 2 = 10 points
- Task 3 = 20 points

In this section, we will use logisim to develop the circuits. You may download logisim directly on your computer if Java is installed correctly on your laptop. Otherwise you can create new container with Java and logisim in it. Logisim can be downloaded using the site below: `http://www.cburch.com/logisim/download.html`

Task 1: Once you are able to successfully access logisim, develop the circuits for the following expressions:

- **C1:** Develop a 5-bit adder using 5 full adders. You should develop all the full adders from scratch, similar to the circuits developed in class, and connect those full adders to get the 5-bit adder working. Develop the circuit using 5 bits inputs for input1 and input2 respectively. You can use a splitter for capturing the input and for displaying the output on the HEX display. For example, you should test the correctness of the circuit using the test cases below:

  First: 0 0 1 0 1 [5]

  Second: 1 0 1 0 1 [21]

  Output: 1 1 0 1 0 [26]

  It is recommend that you test more cases to further verify the correctness of the circuit.

After completing the circuits and testing it, save the files using the file name `C1.circ`.

Task 2: Once you are able to successfully access logisim, develop the circuits for the following expressions:

- **C2:** Develop a 4-bit Two's complement using 4 inputs and the output. Develop the circuit using 4 bits inputs for input1 and input2 respectively. For this part, you may use the builtin full adder directly. If you like to develop using your own custom full adder, it is acceptable to do it that way. You can use a splitter for capturing the input and for displaying the output on the HEX display.

  Input: 0 1 0 1 [5]

  Output: 1 0 1 1 [-5]

  It is recommend that you test more cases to further verify the correctness of the circuit.

After completing the circuits and testing it, save the files using the file name `C2.circ`.

Task 3: Once you are able to successfully access logisim, develop the circuits for the following expressions:

- **C3:** Develop a 4-bit subtractor using 4 inputs and the output. Develop the circuit using 4 bits inputs for input1 and input2 respectively. For this part, you may use the builtin full adder directly. If you like to develop using your own custom full adder, it is acceptable to do it that way. You can use a splitter for capturing the input and for displaying the output on the HEX display.

  First: 1 0 1 0 1 [21]

  Second: 0 0 1 0 1 [5]

  Output: 1 0 0 0 0 [16]

  It is recommend that you test more cases to further verify the correctness of the circuit.

After completing the circuits and testing it, save the files using the file name `C3.circ`.

An image is displayed in the last page, if you need additional guidance on how to develop the subtractor circuit. Only look at it after you had tried out yourself and unsure how to implement it.

Section 2: Assembly Language Starter

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

- Task 4 = 10 points

In this section, we will use mars to develop our first assembly language programming. You may download logisim directly on your computer if Java is installed correctly on your laptop. Otherwise you can create new container with Java and logisim in it. Logisim can be downloaded using the site below:
`https://courses.missouristate.edu/kenvollmar/mars/download.htm`
Alternatively, you may execute the hello world assembly program in the Docker container, within the src folder, using the commands below:
java -jar mars.jar hello.asm

Task 4: Once you are able to successfully access mars, go through the simple hello world program and get used to the assembly programming. Next, update the calculator.asm program to do sub, mul, and div. For example, if the user provided input is 10 and 2, then the output should be 12 8 20 5 printed in new lines respectively. The output should be displayed based on the add, sub, mul, and div operations. You may use new registers such as t3, t4, t5, and t6 for holding the output of the sub, mul, and div operators. Display the output using a new line. More technical points connected to the technique used in displaying result will be discussed at the start of the lab session.

Section 03 - 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 04 - 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

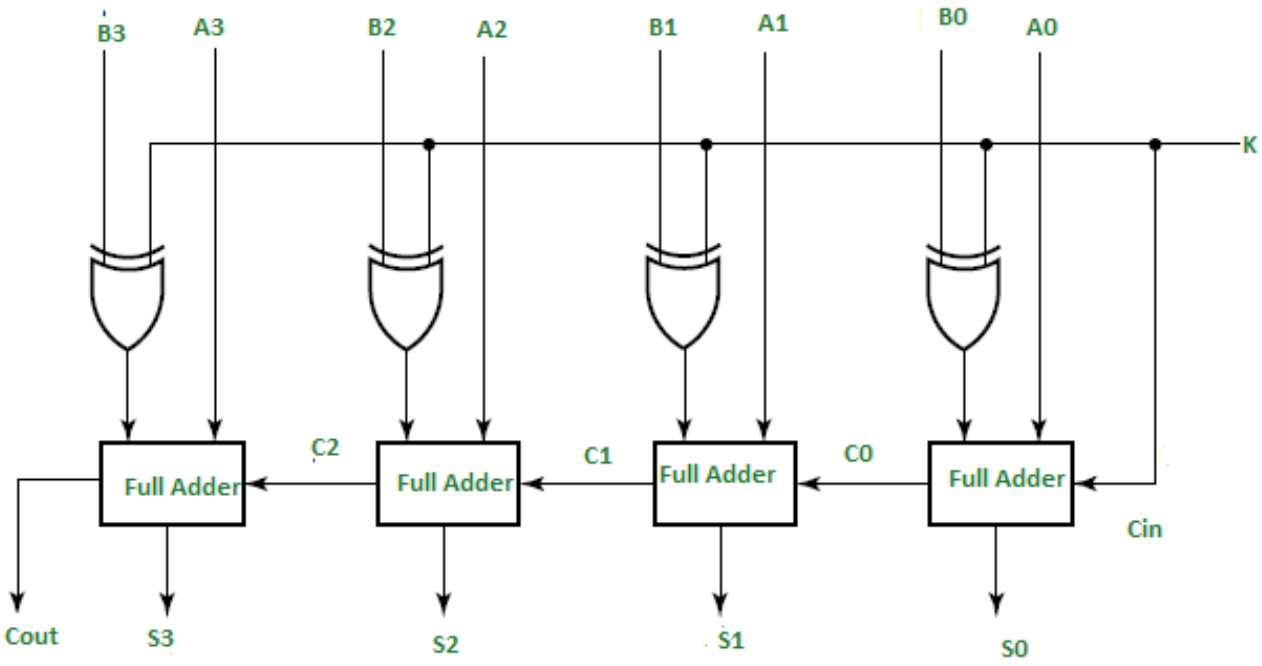
For this assignment, please submit the following to your GitHub lab repository.

1. new version of **C1.circ** file.
2. new version of **C2.circ** file.

3. new version of **C3.circ** file.
4. updated version of **calculator.asm** file.
5. A signed honor code file, named `Honorcode`.
6. 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.


B3
A3
B2
A2
B1
A1
B0
A0
K
Full Adder
Full Adder
Full Adder
Full Adder
C2
C1
C0
Cin
Cout
S3
S2
S1
S0