# CS 2110 Timed Lab 1
# Arithmetic Logic Units

The Best TAs

Spring 2021

## Contents

# 1 Timed Lab Rules—Please Read

## 1.1 General Rules

1. You are allowed to submit this timed lab starting from the moment the assignment is released, until your individual lab period is over. This means you'll have the full **75 minutes** that is reserved for lab—no more and no less (unless you have accommodations or special circumstances that have already been discussed with your professor). Gradescope submissions will remain open but you are not allowed to submit after the lab period is over.

    - You may submit to Gradescope as many times as you wish within your allotted test time period. We will grade your last submission.
    - **Submitting or resubmitting the assignment after this is a violation of the honor code—doing so will automatically incur a zero on the assignment and might be referred to the Office of Student Integrity.**

2. Although you may ask TAs for clarification (private instructor-only piazza post), you are ultimately responsible for what you submit. **The information provided in this Timed Lab document takes precedence.** If in doubt, please make sure to indicate any conflicting information to your TAs.

3. Resources you are allowed to use during the timed lab:

    - Assignment files
    - Previous homework and lab submissions (this includes homework PDFs)
    - Class Notes (Open Net, Open Book)
    - Your mind!

4. Resources you are **NOT** allowed to use:

    - Email/messaging
    - Contact in any form with any other person besides TAs

## 1.2 Submission Rules

1. Follow the guidelines under the Deliverables section.

2. You are also responsible for ensuring that what you turned in is what you meant to turn in. After submitting you should be sure to download your submission into a brand new folder and test if it works. No excuses if you submit the wrong files, what you turn in is what we grade. In addition, your assignment must be turned in via Gradescope. Under no circumstances whatsoever will we accept any email submission of an assignment. Note: if you were granted an extension you will still turn in the assignment over Gradescope.

3. Do not submit links to files. We will not grade assignments submitted this way as it is easy to change the files after the submission period ends.

## 1.3 Is collaboration allowed?

**Absolutely NOT. No collaboration is allowed for timed labs.**

# 2 Overview

In this timed lab, you will be creating an ALU with four operations. This ALU will take in **two 8-bit inputs** and **one 2-bit op code**. It will output **one 8-bit output**.

## 2.1 Tasks and Strategy

- Set up your MUX for the ALU output.
- Build your first operation circuit, and connect it to the MUX. Then submit it to Gradescope.
- Build your second operation circuit and connect it to the MUX. Submit it to Gradescope.
- Rinse and repeat (do operations 3 and 4).
- *Hint: After you complete each operation, you may want to save a local copy of your file as a backup. That way, if you have a problem later on, you can revert to the prior working version quickly.*

## 2.2 Allowed Components

When building this ALU, you may only use:

1. basic logic gates (AND, OR, NAND, NOR, NOT, XOR),
2. decoders,
3. multiplexers,
4. the built-in Circuitsim adders (**NOT** the built-in subtractors),
5. splitters,
6. wires,
7. tunnels,
8. constants,
9. input pins,
10. output pins

**IMPORTANT NOTE 1:** YOU DO NOT NEED TO BUILD THE GATES OUT OF TRANSISTORS. PLEASE, FOR YOUR OWN SAKE, DON'T DO IT. USE THE BUILT IN GATES.

**IMPORTANT NOTE 2:** You're allowed to use CircuitSim's **default, built-in adders** (in the Arithmetic tab). So please don't try to make your own adders, just use that one. You're not allowed to use anything else from the Arithmetic tab (don't try to use a subtractor; if you need a subtractor, you will need to create your own using the above-mentioned components).

# 3 Instructions

## 3.1 CircuitSim Information

For this assignment, you will be using CircuitSim. The version is the exact same as the one used in Homework 2 and 3, and can be found on the Docker image provided for this class. To ensure you are on the correct version, check to see if the title bar says "CircuitSim v1.8.2 2110 edition". **If your file does not open in this version of CircuitSim you will receive a 0**. All changes should be made in the ***tl1.sim*** file. Do not move or rename any or the input or output pins.

## 3.2 ALU Components

You will create an 8-bit ALU with the following operations, using any of the gates listed above. All numbers should be interpreted as 2's complement.

| | | |
|---|---|---|
| 00. | (A − B) / 2 | `[(A - B) / 2]` |
| 01. | signOfB | `[B < 0 return -1, B == 0 return 0, B > 0 return 1]` |
| 10. | merge | `[Ex. A = 00000000, B = 11111111, output = 00001111]` |
| 11. | logicalImplies | `[A -> B, see truth table below for more information]` |

Below are descriptions to elaborate more on the problems above. If you're still struggling to comprehend what the question is exactly asking after reading below, please ask one of your TAs for clarification.

- In the **(A - B) / 2** operation, note that for (A - B), the test cases will **ALWAYS** have an even number as a result of this operation. Also, overflow from (A - B) is not to be mitigated in any way. For example, for (-100 - 50) / 2, the answer will NOT be -75. Rather, it would be (-100 - 50) / 2 = (106) / 2 = 53.
- For the **signOfB** operation, you need to return the sign of B. If B $<$ 0, as in if B is negative, return -1. If B == 0, return 0. If B $>$ 0, as in B is positive, return 1.
- Notice that signOfB depends solely on the `B` input. **It should NOT rely on `A` being a particular value.**
- **merge** should return an output where the 4 most significant bits (leftmost bits) correspond to the 4 most significant bits of A, and the 4 least significant bits (rightmost bits) correspond to the 4 least significant bits of B. For example, if A = `10111111` and B = `01011010`, the output should be `10111010`. In the output, the first 4 bits `1011` are from A, and the last 4 bits `1010` are from B.
- For the **logicalImplies** operation, you'll be implementing the logical implication table, which has been included below for reference. You should apply this operation to all 8 bits using A and B.

| A | B | Output |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- This ALU has two **8-bit** inputs for `A` and `B` and one **2-bit** input for `OP`, the op-code for the operation in the list above. It has one **8-bit** output named `out`.
- The provided autograder will check the op-codes according to the order listed above (A − B) / 2 (`00`), signOfB (`01`), etc.) and thus it is important that the operations are in this exact order.

# 4 Checking your work

You can locally check your grade by navigating to the directory containing `tl1.sim` and running

```
java -jar tl1-tester.jar
```

# 5 Deliverables

Please upload the following files onto the assignment on Gradescope:

1. `tl1.sim`