Introduction

This assignment is a *group* assignment. If I, in the room, do not **hear** you interacting with the other students, you are doing it *wrong*!

Assignment Goals

Learning Outcomes After completing this group assignment, each student is expected to be able to

- *Implement* a `PC` with a `Register` and combinatorial logic.
- *Discuss* simpler `PC` variants (and see their implementation).

Procedure

Get out paper for a *single* turn-in at the end of class. Copy enough of each question so that the paper could stand alone as a study guide.
Assign Roles. Manager, **Recorder**, **Reflector**, **Speaker**.
Answer these questions.

1. What is the difference between *combinatorial* circuits and *synchronous* circuits? Answer in terms of when **outputs** change relative to when **inputs** change.

2. Given an unlimited number of sequential circuits (including plexers and adders) and one (1) 32-bit `Register`, construct a `SimpleProgramCounter` that has a 32-bit output (its current value) and increments by 4 on every clock tick.

3. What makes the `SimpleProgramCounter` less useful than a proper `ProgramCounter`? What *can't* it do?

4. Modify the `SimpleProgramCounter` to a `PausableSimpleProgramCounter`: it has an input, `run`, and it increments only on clock ticks while `run` is high.

5. Extend the **`SimpleProgramCounter`** to a `LoadableSimpleProgramCounter`: it has inputs: `D` with a new value to load; `en` to *enable* loading the new value. On a clock tick, if `en`, latch `D`, otherwise latch `Q` plus 4.

6. Extend the `LoadableSimpleProgramCounter` to a `ProgramCounter`. Inputs are extended with a `reset` flag: if `reset`, latch 0.

reset	en	Latch
0	0	Q+4
0	1	D
1	0	0
1	1	0

7. Which flag, `en` or `reset`, has *priority* in the `ProgramCounter` circuit?

   (No need to write the whole circuit.) *How* would you reverse the priority between them?