In this lab you will work on:

1. *Finite State Machines*
2. *Mealy Machines*
3. *State Diagrams*

Final Lab Project:

Congrats on making it to the final CMPEN 270 Lab Project! This is a **two-week lab** with separate parts due each week. Part I the state diagram, will be due at the end of your lab section during the week of 12/1-12/3. Part II the EDA Playground portion will be due at the end of your lab section during the week of 12/8-12/10.

## Game Boy Problem Description:

For the Final Lab Project, you will be creating a Game Boy game! You'll be creating this game as a Mealy Machine which you should know from lecture means that the output depends on both state and input! You can use the same FSM set up that we used in the previous lab and showed in the video example. The Game Boy program will be slightly different since it is a Mealy Machine.

Your Player will come in contact with multiple different obstacles and will need to be able to perform the following actions to safely make their way through the game!

They will need to be able to run to advance forward in the game, fight by kicking and punching to defeat any enemies they come across, jump to get over tree stumps on the ground, and duck to avoid any bee hives in the trees.

This is what your Game Boy looks like with 4 different buttons: Fight, React, A, B
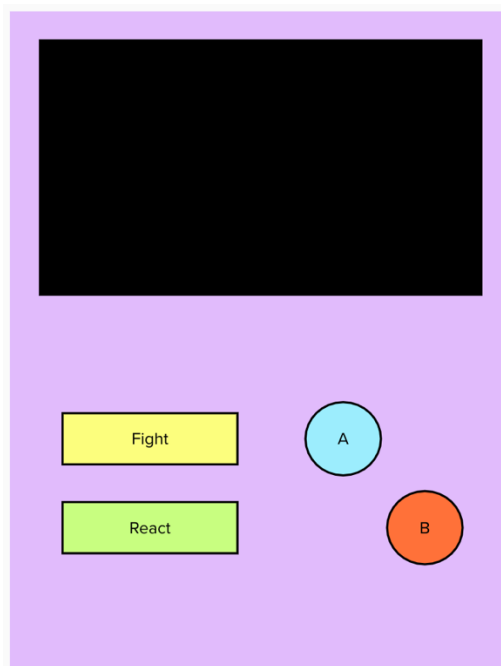
Your player will have 3 different modes:

1. Idle: In this mode your player is able to run using A or B or move to Fight mode or React mode by clicking their respective buttons
2. Fight: In this mode your player is able to kick using A or punch using B or move back into Idle mode by clicking the Fight button again
3. React: In this mode your player is able to jump using A or duck using B or move back into Idle mode by clicking the React button again

Notes: Your player cannot go from Fight mode to React mode and vice versa.

Part I - State Machine Diagram:

Due 12/1-12/3

In Part I you are required to create a state machine diagram that follows the problem statement above and requirements below. This diagram is necessary to complete Part II.

Project Requirements:

Mealy Machine Diagram

*   The output is your players action, so there are five possibilities: Kick (0), Punch (1), Jump (2), Duck (3), Run (4) *the corresponding numbers should be what your output is set to for each of the actions but in your diagram, you may use the action names*
*   Your Game Boy has four different inputs: Fight (0), React (1), A(2), B(3) *the corresponding numbers are the actual values of your input but in your diagram, you may use the input names*
*   depending on the input and state you will either move to a new mode or stay in your current mode and perform an action
*   whenever you are in Idle mode or move to Idle mode your action should always be Run
*   When transitioning from Idle mode to another mode we don't care what your action is
*   When Reset = 1 your player should be in Idle mode

Part I deliverables - Once you have completed the state machine diagram for this problem statement make sure to upload a picture to canvas under the Final Lab Part I assignment and/or show to your TA to get graded during lab.

Part II - Verilog:

Due 12/8-12/10

EDA Playground starter files: https://edaplayground.com/x/EnjE

Now that you have created a state machine diagram for the problem description you need to implement it in EDA Playground using Verilog. If you did not do your state machine correctly your TA's will be sharing the correct diagram during this week's lab in order for you to still be able to complete Part II correctly.

Project Requirements:

Design.sv

- You will have a module Design.sv that will be your top level module and act as your Mealy Machine
- Design.sv will have three inputs clock, x, and reset and one output z
- Your input x will be 2 bits and your output z will need to be 3 bits
- Your will need to output 0 for a Kick action, 1 for a Punch action, 2 for a Jump action, 3 for a Duck action, and 4 for a Run action *(it is recommended that you use the parameter assignment for these to make your program easier to read)*
- This module should follow the state diagram created in Part I, it should contain two always blocks: one that determines the next state and another that determines the output
- z can be set to 3'bx if we don't care about its value
- when reset is 1 the player should be in Idle mode

Input output format:

1. Once again, the testbench is completed for you, you do not need to change anything on it.
2. If you have completed the project successfully the following output

```
BEGIN GAME

Run
    __0
   / /\_,
 __/\
    /_

INCOMING ENEMY GET READY TO FIGHT ..

ATTACK!

Kick
\_0__
   \__      __
   |  \__  /00\
  /      /   \

Punch
      _____
 ---'    ____)
      (_____)   __
      (_____)  /00\
 ---.__(___)  /   \

Kick
\_0__
   \__      __
   |  \__  /00\
  /      /   \


Punch
      _____
 ---'    ____)
      (_____)   __
      (_____)  /00\
 ---.__(___)  /   \

ENEMY DEFEATED .. CONTINUE TRAVELING

Run
    __0
   / /\_,
 __/\
    /_

ENTERING JUNGLE GET READY TO REACT TO DANGERS ..

INCOMING TREE STUMP

Jump
     \0
      |\_
      /-\
   /      \
       __
      |__|
```

```
INCOMING BEE HIVES ABOVE

Duck
(____)
 (__)
  (_)

  __0
 /  \

INCOMING TREE STUMP

Jump
    \0
     |\_
    /-\
   /     \
     __
    |__|

EXITED JUNGLE SAFELY .. CONTINUE TRAVELING

Duck
(____)
 (__)
  (_)

  __0
 /  \
```

Run
```
    __0
   / /\_,
 ___/\
     /_
```

```
***** WINNER ******
```

If you see:

XXXXXXXX GAME OVER XXXXXXXXX

This means your player did not make it and your program failed.