**University of Michigan – Dearborn**

**CIS 310– Computer Organization and Assembly Language**

**Assignment #2**

**Memory, Registers, and Program Counter for the CPU**

Hailey Gumbs

[ghailey@umich.edu](mailto:ghailey@umich.edu)

Yahya Daoud

[yydaoud@umich.edu](mailto:yydaoud@umich.edu)

Ciaran Grabowski

[ciarang@umich.edu](mailto:ciarang@umich.edu)

October 30th, 2024

**Table Content**

Contents

[**Program Counter 3**](#_heading=h.gjdgxs)

[Test Cases 3](#_heading=h.3znysh7)

[**Register 3**](#_heading=h.c6wq423y78kg)

[Test Cases 3](#_heading=h.t9baixqxb9cq)

[**4-bit Register file 3**](#_heading=h.xrfskeyeihq)

[Test Cases 3](#_heading=h.lnkz83dfapxk)

[**16 x 4 bit Memory and RAM 3**](#_heading=h.ocf2exa57h1e)

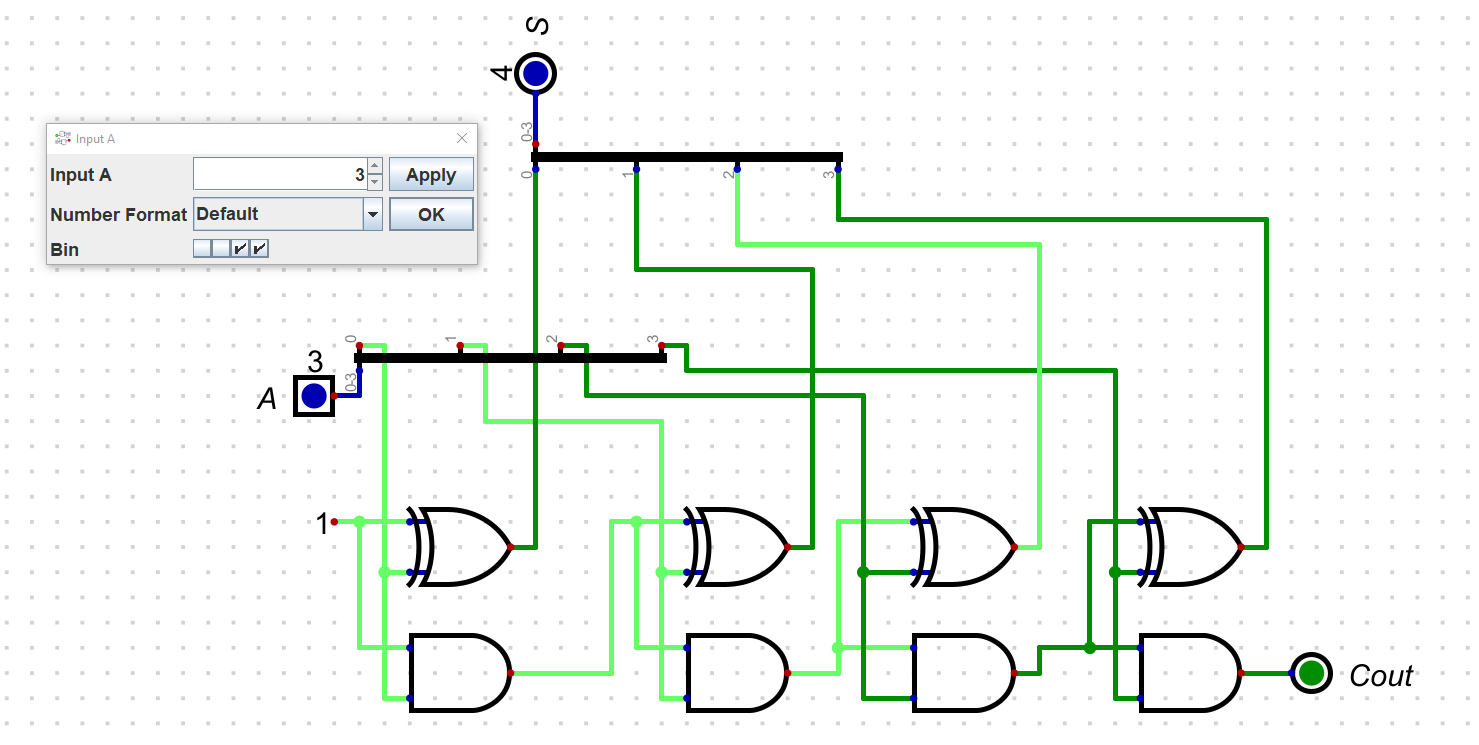
[Test Cases 3](#_heading=h.omnzp8wzkhgy)

# Program Counter

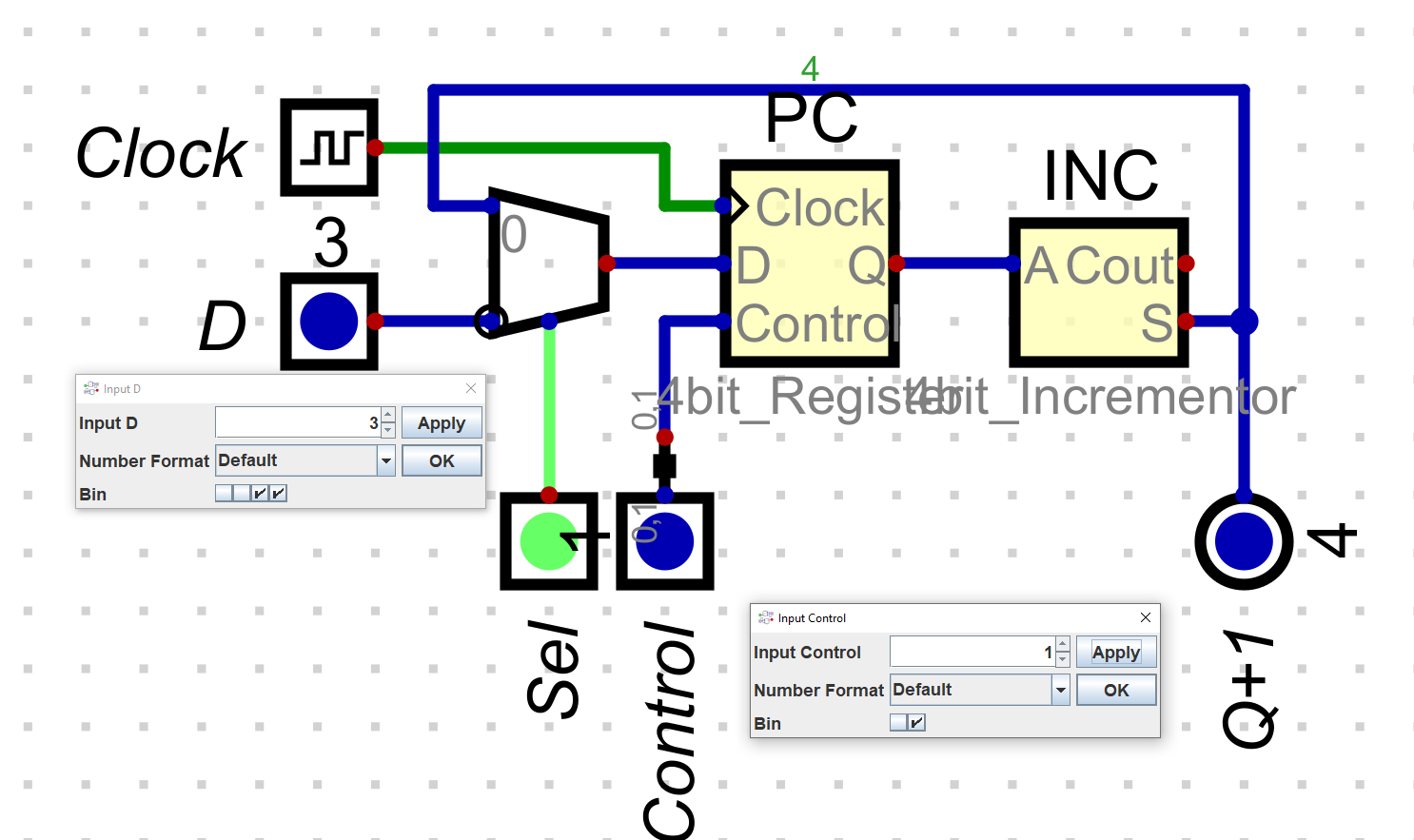
The program counter increments the instructions so the CPU can do the instructions in sequential order. It takes in a clock to synchronize the circuit, an integer that can be zero for the start of the instructions or loads in another integer to execute instructions further down. This input is fed into a MUX control by a selector (another input) which goes to a register to hold onto the number. The number is fed into our incrementing mechanism that adds one. That result is fed back into the MUX at the start so the process can loop as instructions are executed.

## Test Cases

### Incrementing Mechanism



### Program counter



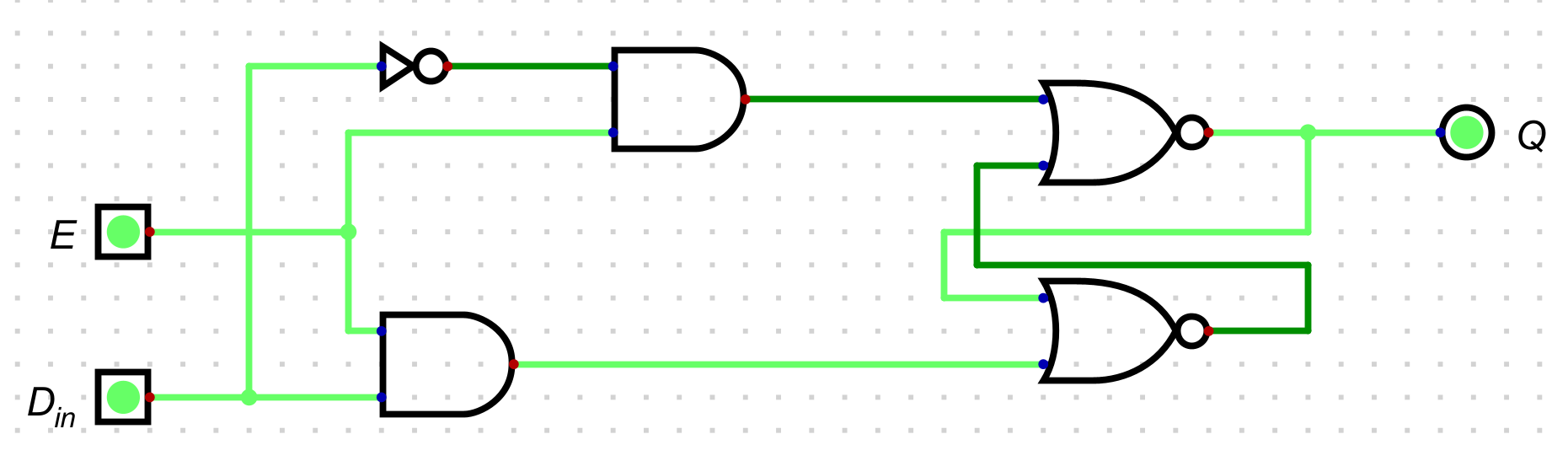
# Register

Our 4-bit register is just 4 1-bit D flip-flops that are controlled by a MUX. This MUX takes in 1 input with the other two being inactive, this is meant to control whether the register is read or written. A clock synchronizes the flip-flops.

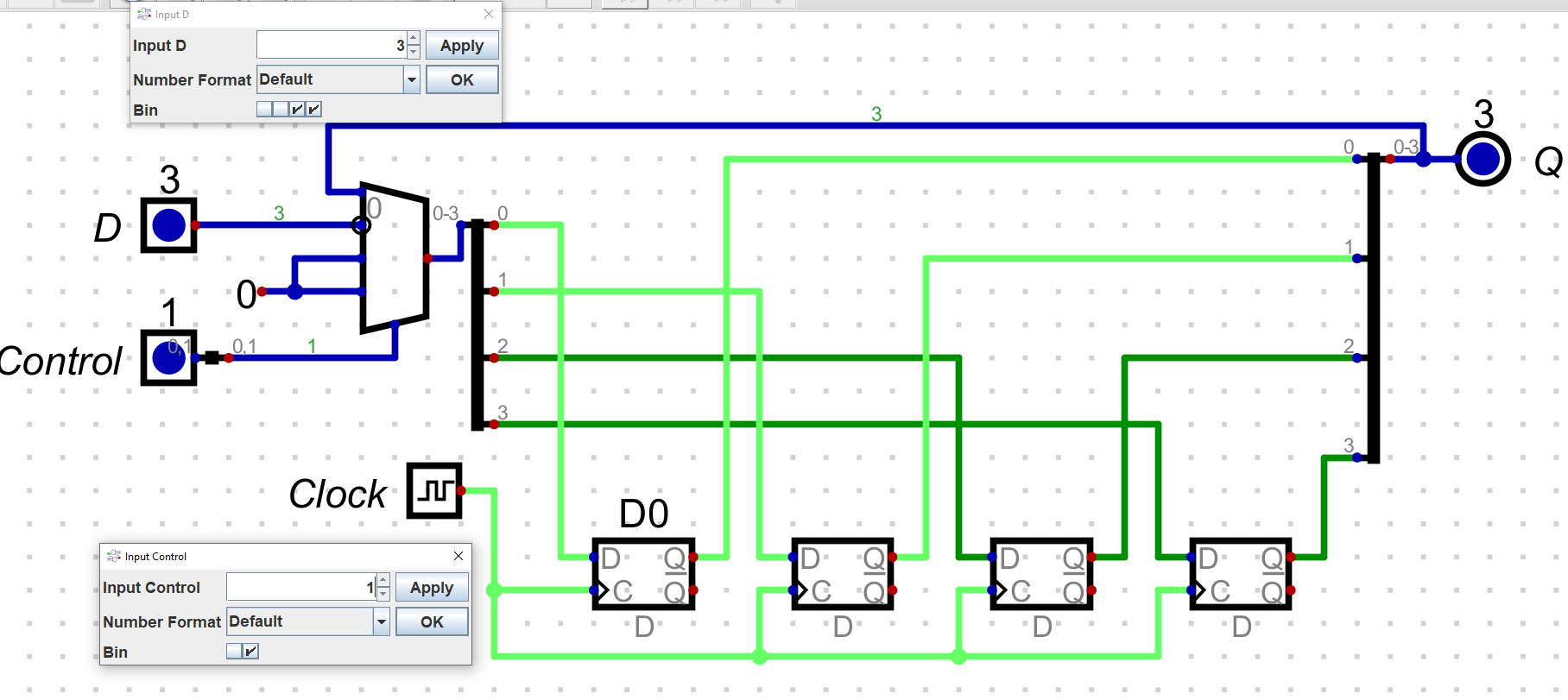
This is an updated version based on the one we did in class. Our original used transistors(drivers in Digital Simulator) to control this mechanism.

## Test Cases

### D-latch



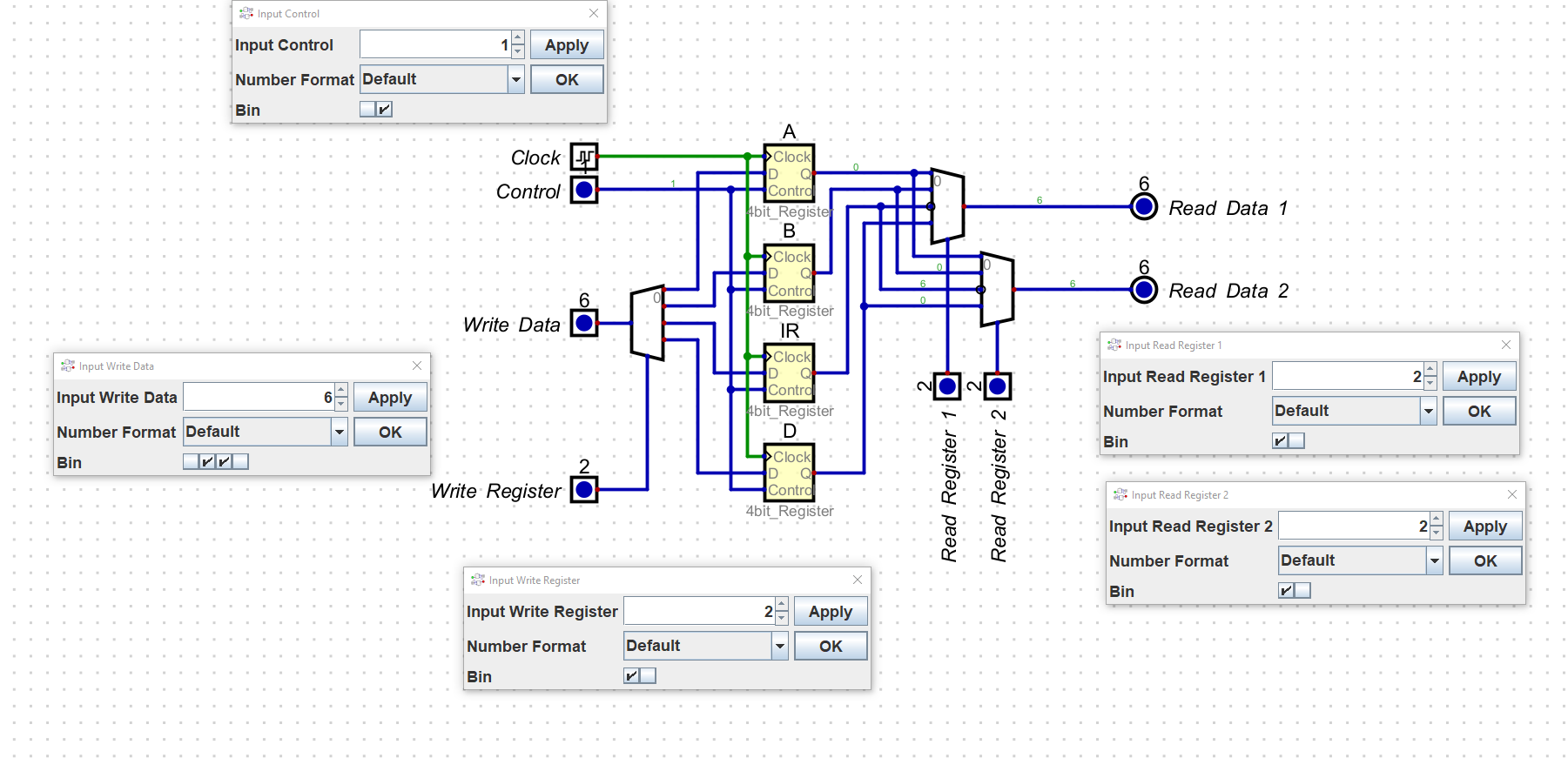
### 4-bit Register



# 4-bit Register file

The 4-bit register file is 4 separate registers connected by 1 MUX and 2 deMUXes. The MUX at the start takes in data to write into the registers and a selector picks which register takes that info. The deMUxes select which register is being read from.

## Test Cases



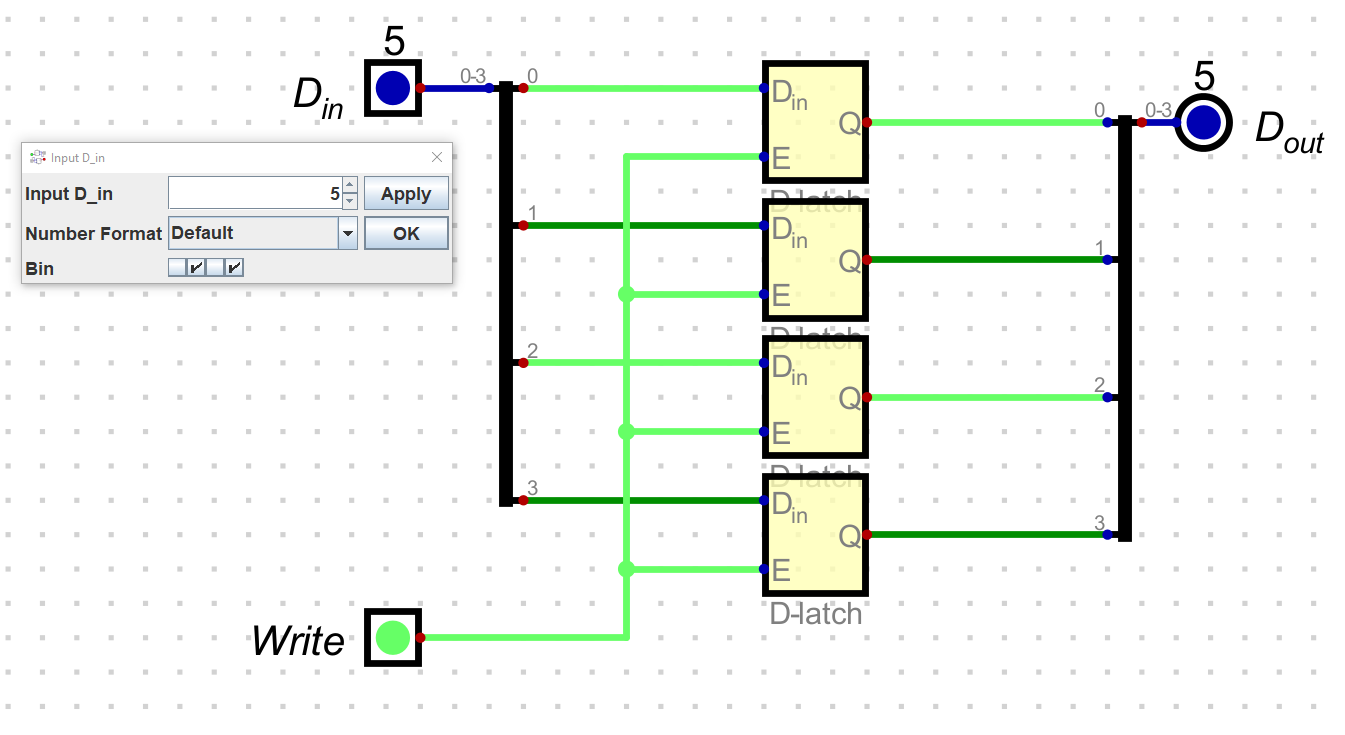
# 16 x 4 bit Memory and RAM

While it is large and appears strange, the 16 by 4-bit memory is 16 individual 4-bit memory circuits connected by a MUX and deMUX that take in the address as the selector. Data is fed into the registers as well.

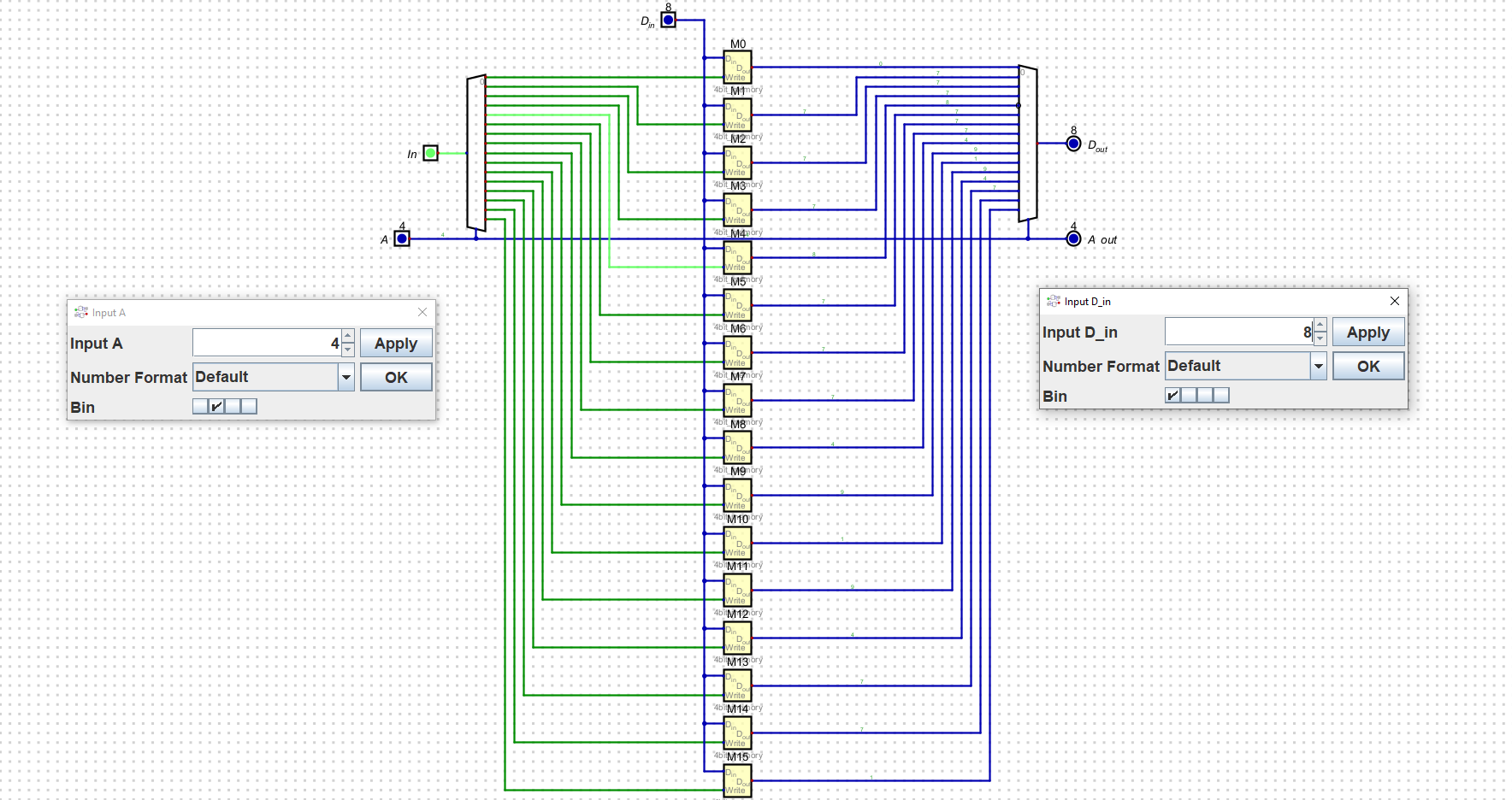
For clarity, we created a separate circuit, the RAM, in order to facilitate the access of information from the memory module.

## Test Cases

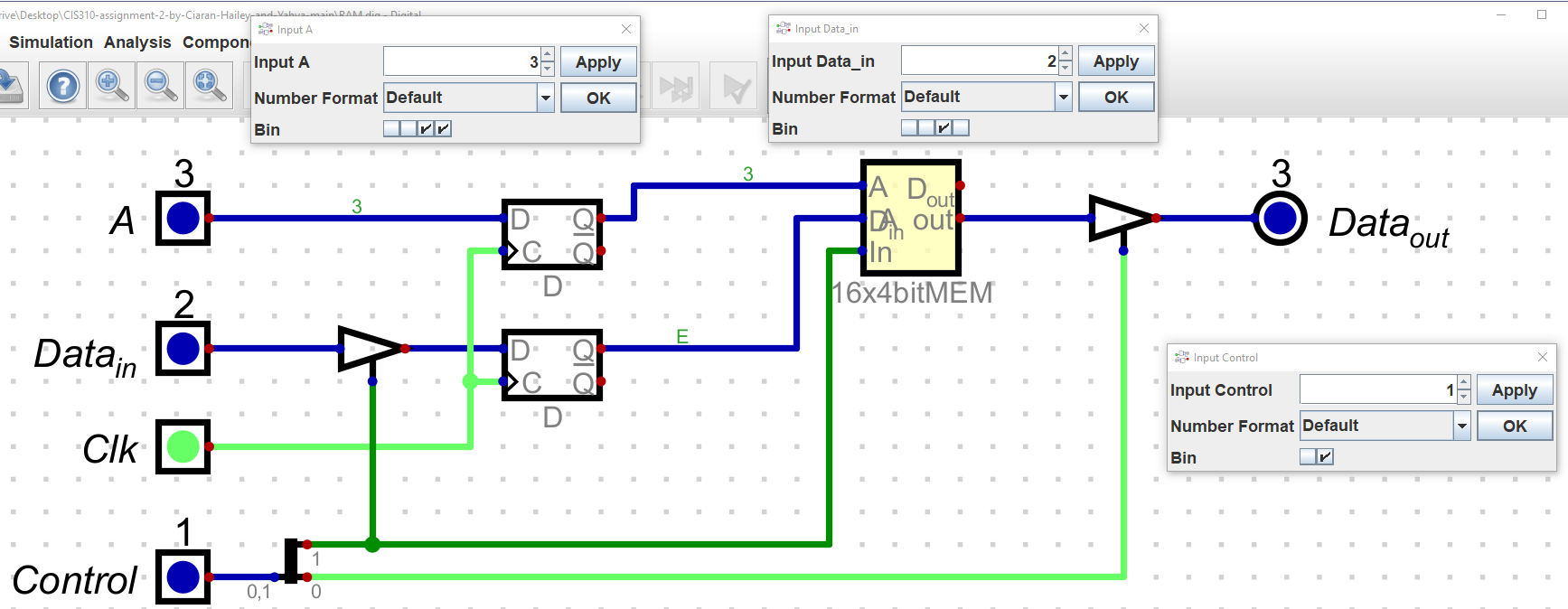
### 4bit memory



### 16 x 4bit memory module



### RAM



Collaboration and Reflection:

This assignment led us to finish using a process similar to the first: dividing and conquering. This time, we didn’t need to go back as much since the biggest struggle parts were addressed in class. Our biggest challenges this time were the Program Counter and the Instruction register. Finding a good way to increment was a struggle to ensure the output reflected our goals. We had a good idea of what to do with the instruction register but struggled with the decoding. Thankfully, we didn’t need to address it immediately for this project.