Computer Architecture

PART A. Using LOGISIM, construct a shift-register of 8 bits. The teaching point here is two-fold: (a) shifting binary patterns left or right is a function available in an ALU, and (b) shifting left really helps in performing the multiplication operation, because, in binary, the partial products are images of the number being multiplied *but shifted left an appropriate number of bits.*

This is perhaps best explained by example:

Consider:

```
10101  *
  101      ←multiplier
_______
1010100    ←partial product, same as number being multiplied but shifted left two places
 000000
  10101    ←final partial product, no shift this time
_______

_______
```

Final answer is the sum of the partial products, as you would do when in decimal.

1. Drag 8 "D-type" flip flops onto the canvas, and moving from left to right, connect the "Q" output of the left flip-flop to the "D" input of the one to its right, forming a chain. We are actually going to ROTATE RIGHT to avoid having to re-input data when observing the behavior of this device, so connect the "Q" output of the final right-most flip-flop back to the "D" input of the first left-most flip-flop.
2. Create a common "Clock line" *driven manually by a single push button.*
3. On each flip-flop, on the south edge, you will find three (3) inputs – the left one is a preset over-ride and when pulsed will set the flip-flop to a "1" regardless of the clock. Connect a pushbutton to all of the flip-flop presets. The right input is a reset over-ride and when pulsed will set the flip-flop to a zero "0", regardless of the clock. Use these push buttons to set the register so that it is holding the number 16 in binary.
4. Manually and slowly apply clock pulses via your pushbutton on the clock line. Observe that each pulse divides the value by two. If we were shifting left, the value would be multiplied by 2, and would be able to form the partial products in a multiplication, as explained above.

PART B. ADDRESSING CONCEPTS. In LOGISIM, on the same canvas, drag a multiplexer on to the circuit. Set the "data select" number to three (3), and "Include enable" to 'no'. You should see 8 inputs on the left edge.

1. Connect pushbuttons to each of the inputs as a first step to understanding addressing.
2. Drag a splitter onto the canvas. Set "Fan-out" and "Bit width in" both to three (3); set "Facing" to left-handed. Connect the upper thick end of the splitter to the address lines of the multiplexer, and connect 3 pins to the bottom inputs of this splitter.
3. Put an l.e.d. on the output of the multiplexer.
4. Simulate. Observe that the value on the address **determines which of the 8 inputs is routed through to the output.** Try several addresses, and notice that **only** the addressed input line of your set of push-buttons affects the output – the others have no effect. It is not co-incidence that the other name for this device is "*selector*". **Memory addresses work in the same way**, so you have experienced the concept of addressing, something which we will cover in class. Also we will be using multiplexers in the three-bus CPU architecture.