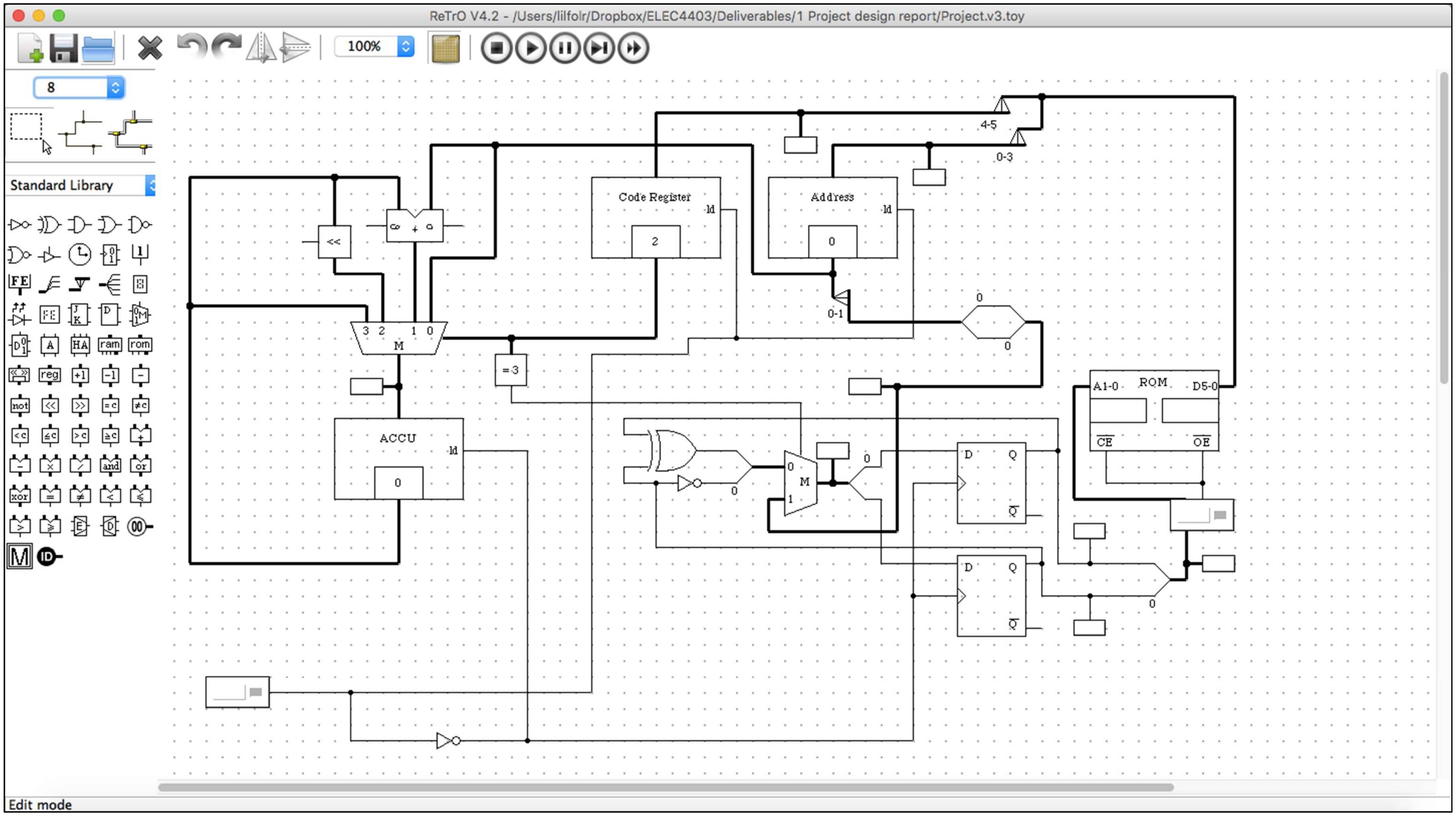
ELEC 4403 - Project Design Report

Table of Contents

Who did what (duties)

Task	**Persons done by**
Planning - Retro	Leighton; Wojtek
Planning - Fritzing	Joseph
Construction – ROM	William; Wojtek
Construction – PC	Joseph
Construction – ALU	Wojtek
Construction – Clock	William; Joseph
Documentation – Design Report	Leighton
Documentation – Bill of materials	William
Documentation – User Manual	Leighton
Documentation – Marketing	Leighton
Debugging	Joseph; Wojtek; William

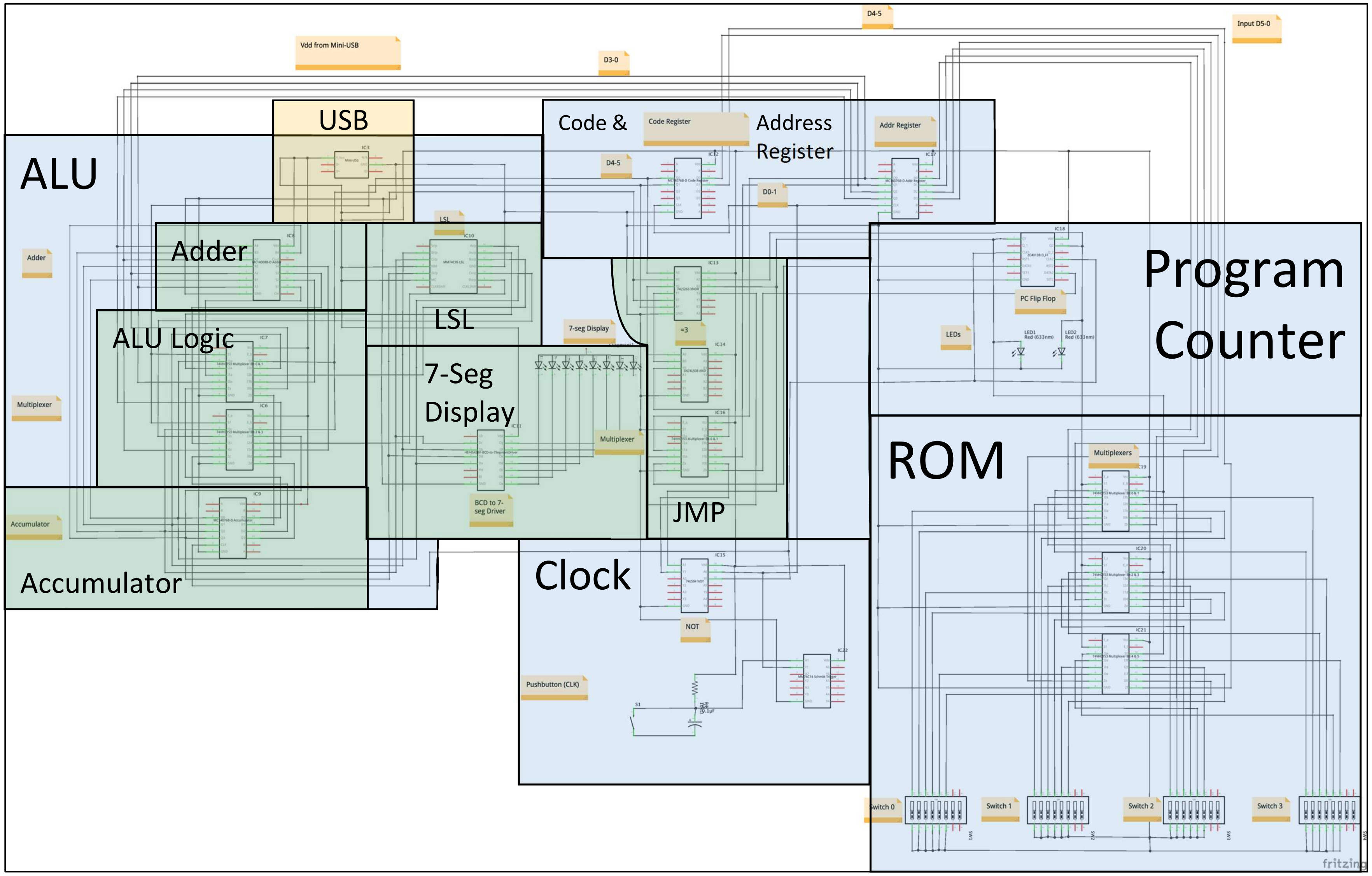
Retro Simulation



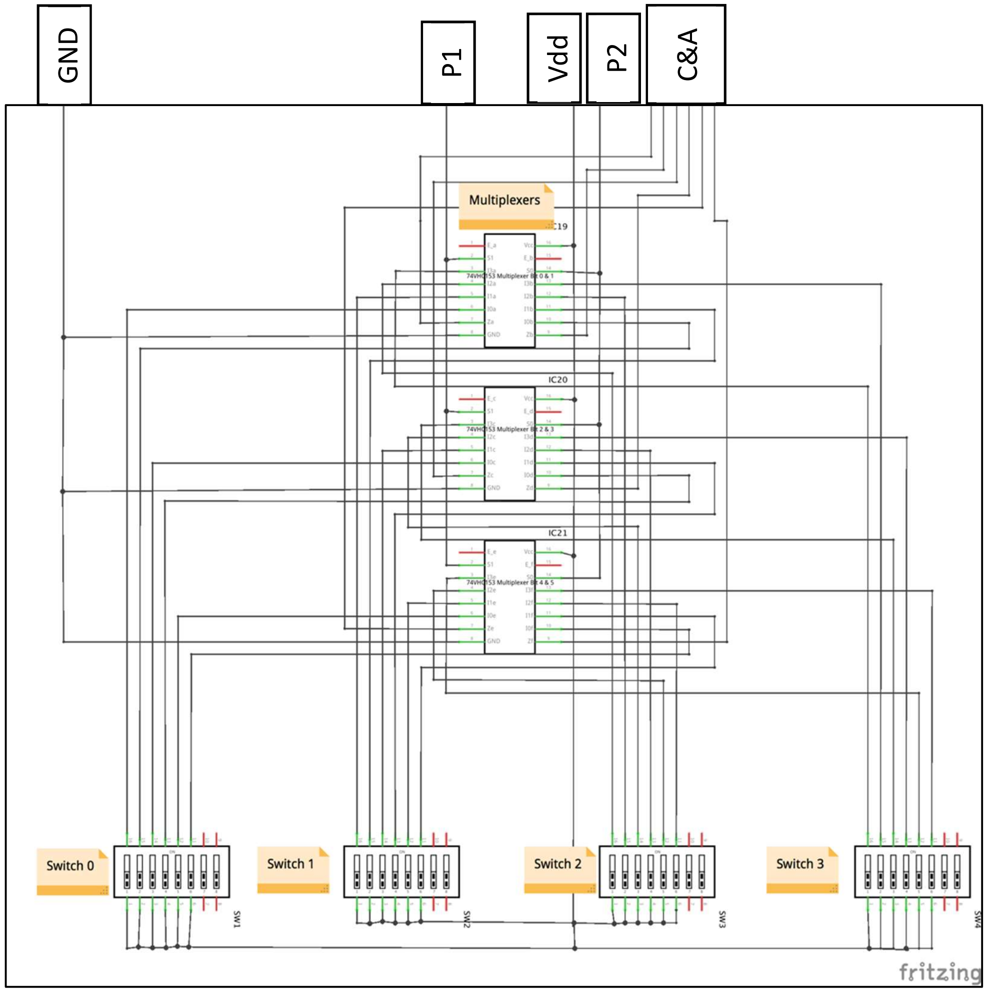
Note: both the retro and fritzing source files can be downloaded from http://elec4403-group111.tk

Circuit Diagram - Fritzing

Complete circuit



ROM



The ROM is represented with 4 dip-switch banks, each containing 6 switches. Hence the ROM has a capacity of 4 words, and a width of 6 bits.
The first 2 bits of each word represent the op code, while the following 4 are for the data, least significant bit to most.
EG: 01 0100 = ADD 2

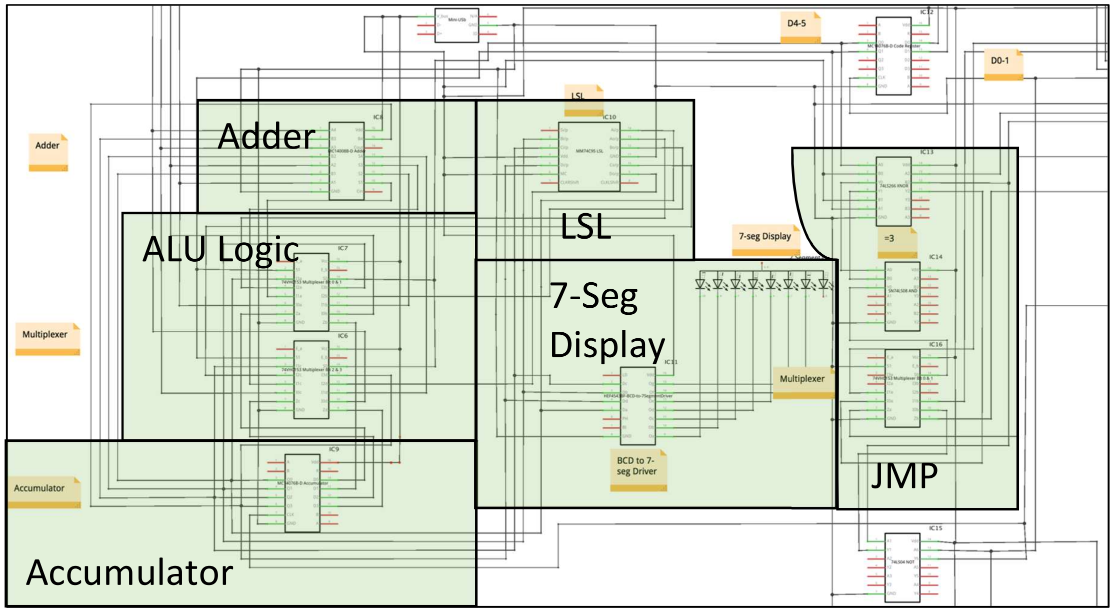
The outputs for the ROM are:

GND	Ground
Vdd	Positive supply
C&A	ROM values address and code register

The inputs for the ROM are the switch positions and.

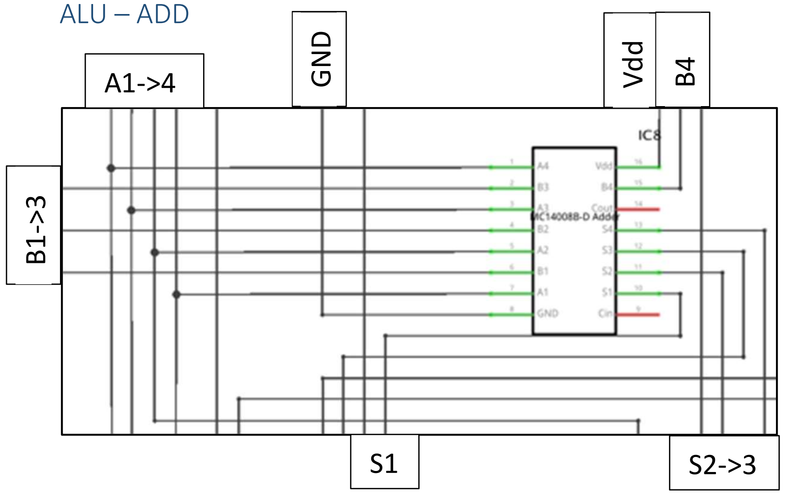
P1 & P2	Program counter value, indicating which ROM word is in use

ALU – Logic



The ALU is a combination of multiple sub-modules, which will each be described below.
The ALU is activated when the second clock signal (triggered with the button is released) reaches the accumulator.
At this stage both the code and address register have been populated with the current op code and data. This means that when the accumulator activates the ALU multiplexers will pick an input to update the it from, depending on the code register's op code.
The Adder, LSL, and Address register (for the load command) feed into these multiplexers as input lines.
Finally, the 7-segment display is connected to the output of the accumulator to display its current value.

ALU – ADD



The adder chip is used whenever the ADD op code is sent. It will take the data in the address register and add it to the data in the accumulator, storing the result in the accumulator.

The chip we've used contains 4 connected adders, with A1; B1 being then input to the first one. A2; B2 are then the input to the second with, along with the carry bit from A1; B1's sum. The results come out of S1->S4, with C_{out} being the final carry bit

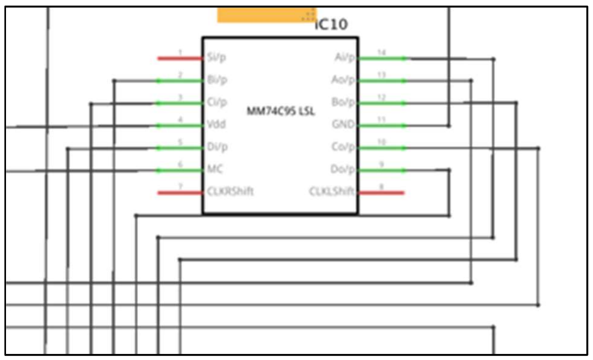
The outputs for the adder are:

GND	Ground
Vdd	Positive supply
A1->A4 & B1->B4	The 8 input bits come from the current values in the accumulator [4] and the current values in the address register. In our case A1->A4 come from the address register, and B1->B4, the accumulator

The inputs for the adder are:

S1->S4	The final sum is first sent to a multiplexer to check if the add operation is active, then if so, to the accumulator
C_{out} & C_{in}	Not used

ALU – LSL



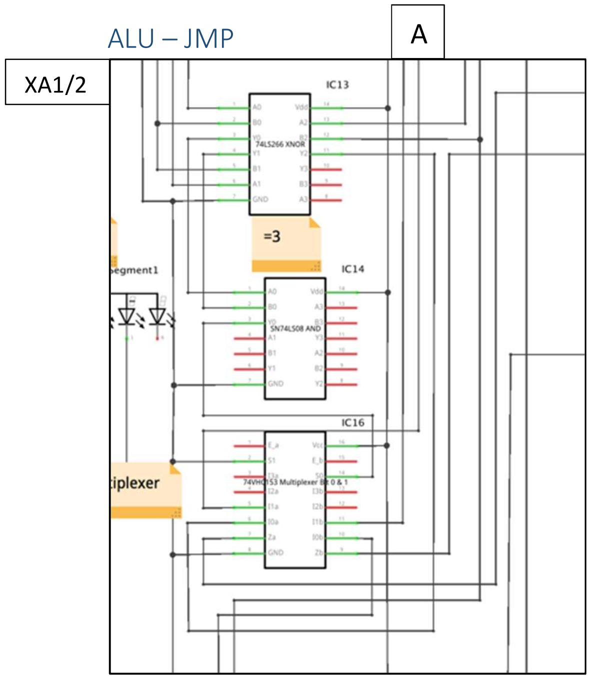
The Logical Shift Left was initially scoped to use a chip like the diagram the left, however it became apparent that implementing this with simple wire crossings was much easier:

In_4	In_3	In_2	In_1	0
	↓	↓	↓	↓
	Out_4	Out_3	Out_2	Out_1

EG: 1001 → 0010

The inputs are the 4 values in the accumulator, and the output is the LSL bits, and is stored back in the accumulator

ALU – JMP



JMP is used when opcode 3 is run, and will update the program counter to the value in the stored address register. The JMP logic is completed using AND; XNOR gates, and a multiplexer.
When the first clock signal comes to the

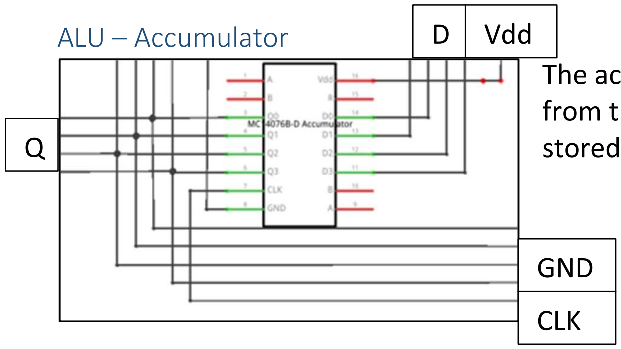
The inputs for the JMP logic are:

XA1; XA2	The opcode from the code register to the XNOR gate; which is XNORed with Vdd Because 3 = 0b11, the XNOR will only return 2 highs for op code 3. This is then passed to the AND gate reducing the 2 highs to 1 high. The 1 high is then the input for the program counter multiplexer
A {2-bits}	When the multiplexer select line is high, the input for the program controller switches from incremental to the value from these inputs – from the address register [first 2 bits]

The outputs for the JMP logic are:

P1; P2	2-bit output to the program counter

ALU – Accumulator



The accumulator is a 4-bit register, activated with the second (NOT) clock signal. The input values come from the ALU logic multiplexers, which, depending on the current op code, either leave the value stored unchanged, or update it.

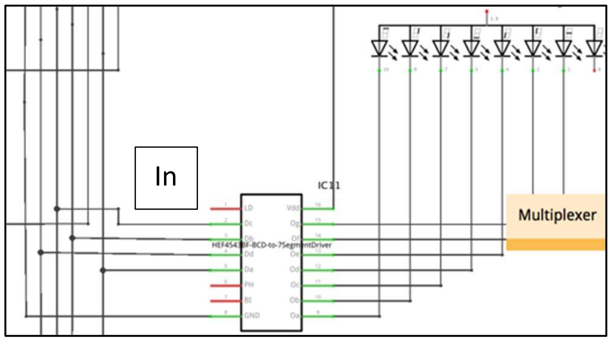
The outputs for the adder are:

GND	Ground
Vdd	Positive supply
Q	The current stored values in the accumulator. These go to the 7-seg display, to the ALU Logic – if an operation that doesn't update the accumulator is use, and to the input of the adder

The inputs for the adder are:

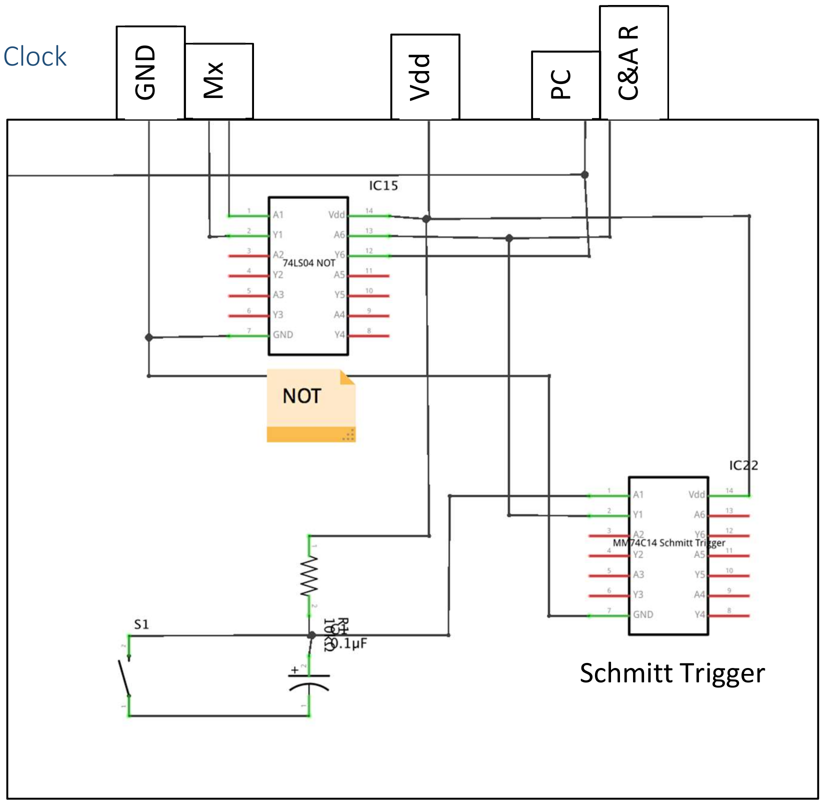
D	The input comes from the multiplexer which will return the new accumulator value when operated on by the active opcode

ALU – 7-Segment Display



The 7-segment display shows the value in the accumulator in hex notation. The display itself (represented by the 8 LEDs on the left) has an input for each segment, and 1 for the decimal point (not used). To convert the number from binary to hex we use a 7-segment driver chip, which takes an input of 4 bits, and produces an output for each of the 7 segments.
The inputs are the 4 bit values from the accumulator, and the output is the value shown on the display.

Clock



The clock is implemented as a push-button switch. To prevent the noise inherent in a button's push being decoded as clock signals a de-bounce circuit was implemented. This is achieved in part by the Schmitt trigger, which adds to the circuits noise resistance.
A Schmitt trigger will have 2 thresholds, high and low [called hysteresis]. If a noisy signal passes the high threshold it will need to then pass the low to be counted as a new pulse. This ensure that noise even around the high threshold will be blocked.

The circuit has a 2 step cycle:
1. load the code and address registers
2. Run the operation and increment the program counter

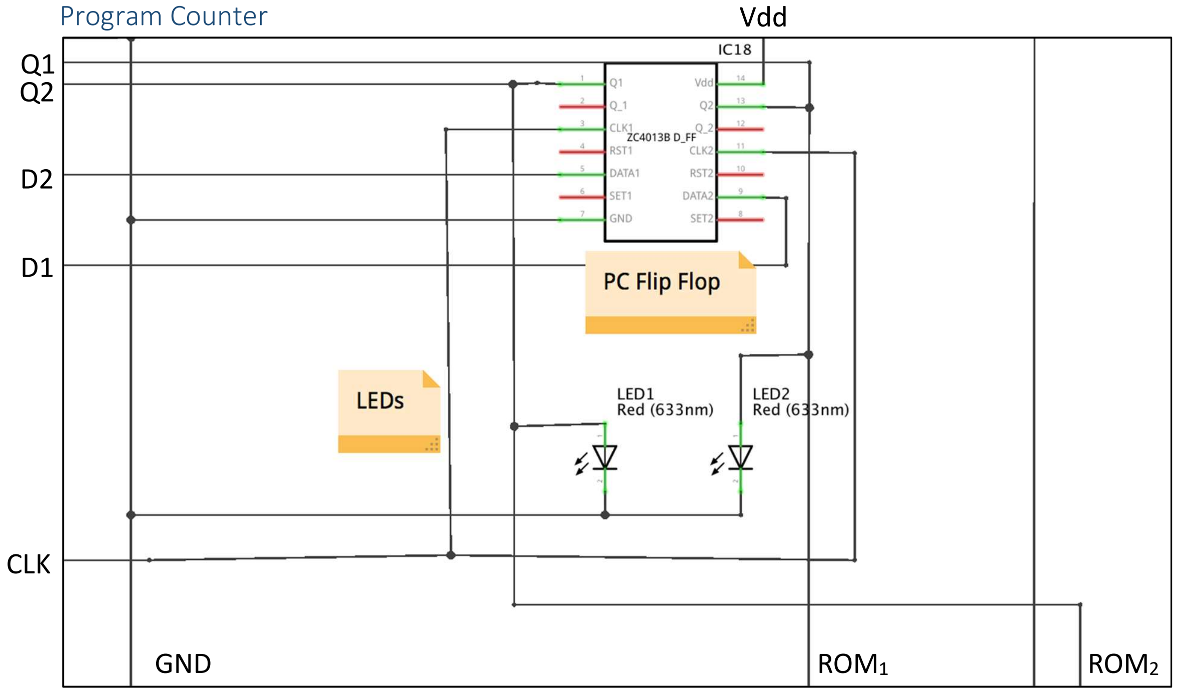
Rather than have the user push the button twice, the second step was wire as a not of the first, meaning it will run when the button is released.

The outputs for the clock are:

GND	Ground
Vdd	Positive supply
Mx	To ROM & JMP multiplexers
PC	To program counter
C&A R	To code and address register

There are no inputs for the clock, baring the user pushing the button to activate CPU's sequential instructions.

Program Counter



The program counter state is represented with a single flip-flop chip, and this value can be viewed via the two LEDs.

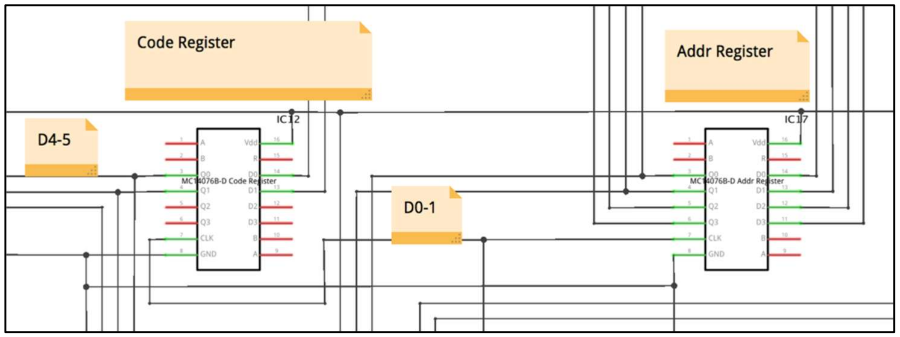
The inputs for this block of the circuit are:

GND	Ground
Vdd	Positive Supply
CLK	Pulse from the clock
D1 & D2	Input from the multiplexer. Either an increment of Q1&Q2, or address of JMP command

The outputs for the program counter are:

ROM$_1$ & ROM$_2$	To the select lines of the ROM
Q1 [=ROM$_1$] & Q2 [=ROM$_2$]	Go back to the input multiplexer so they can be incremented. Not used if JMP is active

Address & Code Register



The address and code register serve to store the current op code and data from the ROM before the ALU acts upon them. Hence the code register is 2 bits while the address register is 4.

The inputs for the code register are:

GND	Ground
Vdd	Positive Supply
CLK	Pulse from the clock
D0 & D1	The op code that comes from the first 2 bits of ROM

The inputs for the address register are:

GND	Ground
Vdd	Positive Supply
CLK	Pulse from the clock
D0 → D4	The data that comes from the last 4 bits of ROM

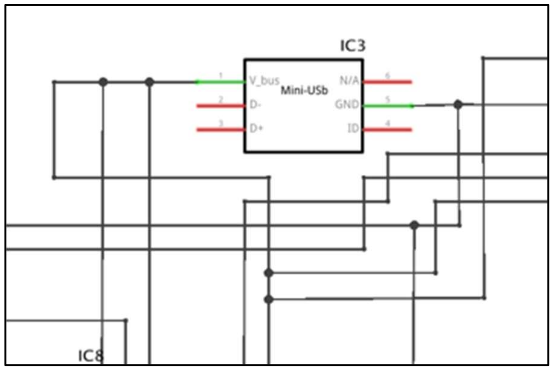
The outputs for the code register are:

Q0 & Q1	The op code in the register that will be sent to the ALU multiplexers

The outputs for the address register are:

Q0 → Q4	The data send to the ALU multiplexers. Note, Q0 & Q1 are also used for the JMP command

USB



The mini-USB connector acts as the power supply for the circuit. The USB circuit has only two outputs, the GND for the circuit, and the 5V supply. These then branch off to power/ground all the multiplexers, registers, flip-flops and other components.