FORMAN CHRISTIAN COLLEGE
(A CHARTERED UNIVERSITY)

Computer Organization and Assembly Language – COMP 300 B

Spring 21

Mid Term Exam

Muhammad Sameed Gilani - 231488347

You should attach the lab / assignment handout as second page of this report.

From third page onwards following headings should be included:

- **Introduction**
  - **Should carry information of all major library functions.**
- **Your logic / algorithm in simple English. Bullet points are appreciated.**
- **Your code**
- **Screen shots of at least three outputs of your code with appropriate inputs.**
- **References**

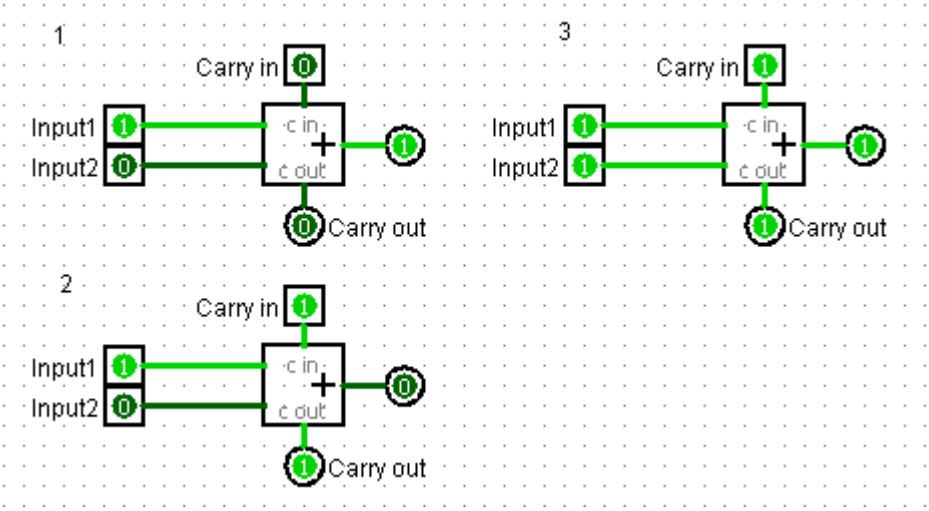
INTRODUCTION

We had a to construct a Data Path, consisting of , an ALU with 4bit operation, a 4x4 Register File, a 16x4 RAM and a Quad 2x1 set of MUXs. A data path is part of the CPU and is used for data processing, where data is primarily stored and transferred from the Register File to the ALU. But additional memory is added in the form of the RAM.

The components used from the Logisim toolbox are,

1. 1bit Adder

It accepts 2, 1bit values and outputs their sum. It has a carry-in input and a carry-out output, to facilitate the addition.



As shown in the 3 samples.

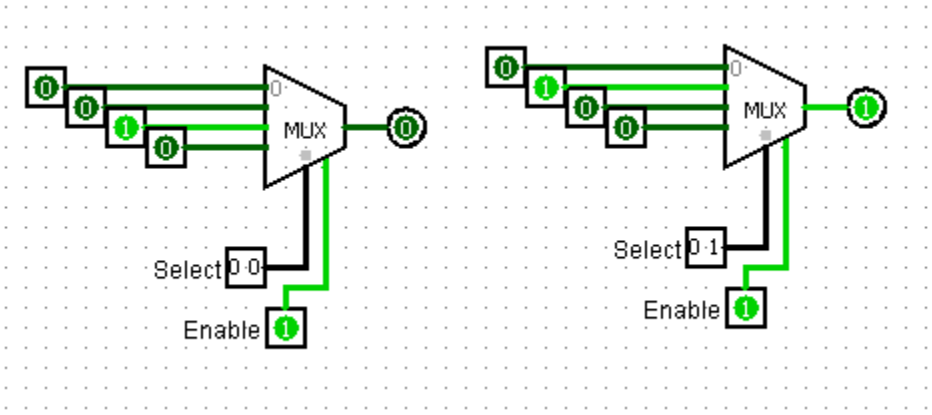
In sample1, 1+0 = 1, with 0 Cout.

In sample2, (Cin)1 + 1 + 0 = 0 with 1 Cout

In Sample3, (Cin)1 + 1 + 1 = 1 with 1 Cout.

2. Multiplexers

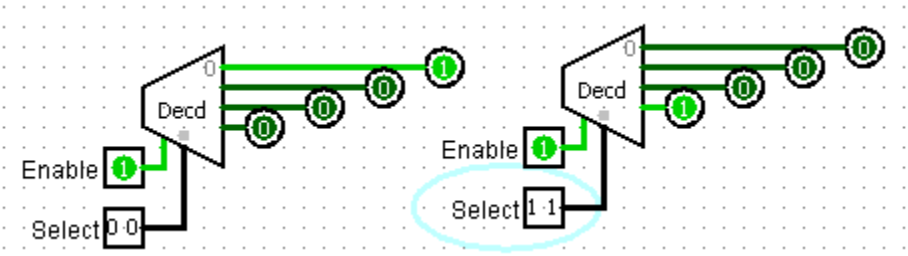
Multiplexers take 2^k inputs and have one output. Using its select input, which is k bits, we select which of its inputs we want to access. It is used to select only one of its inputs to be displayed at the output, as a data selector.



As shown, $2^2 = 4$ inputs and 1 output, but k = 2, select bits.
As in the sample, when the select bit is 00, it is displayed at the output.
And when the select is 01, only it is displayed at the output.

3. Decoders

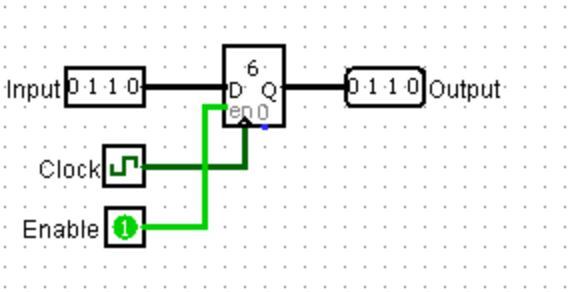
Decoders have 2^k outputs, which are selected with its, k-bits select input. Using the select input, we can choose one of the outputs to be high. This allows us to activate only the 1 particular output, that we require.



As shown, 2^k outputs, with k-bit select.
When the select is 00, the corresponding output is high.
And when it is 11, then its corresponding output is high.

4. Registers

They are used to store and transmit data. With the rising edge of the clock trigger, the input data is stored in the register and the data is available at the output. It keeps the previous value stored, until a new data input is loaded with the clock trigger.



As shown, the input 0110 is loaded and immediately displayed at the output.

5. SR Flip Flop

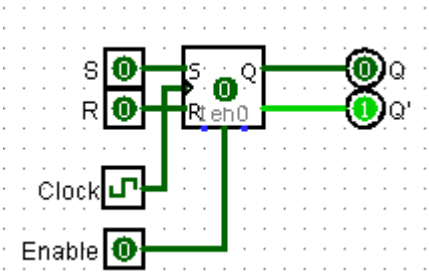
It is a sequential circuit, with 2 inputs, S "SET" and R "RESET" and 2 outputs, Q and Q'. Which follows the basic functionality of,

S=R = 0 → No change state

S=R = 1 → Undefined state

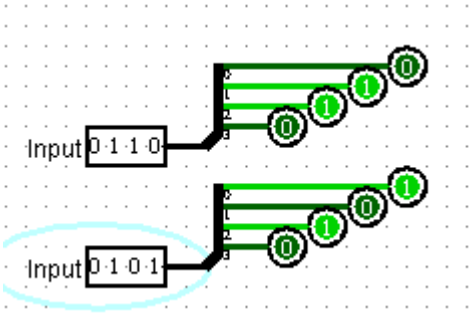
Else, next state follows S.

S	R	Q(t+1)
0	0	Q(t)
0	1	0
1	0	1
1	1	Error



6. Splitters

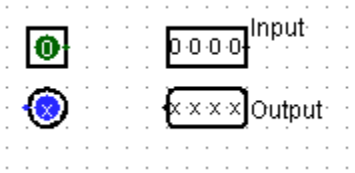
They are used to divide data into single separate bits.



As shown, they are simply dividing the 4 bit data into 4 separate bits.

7. Input and Output Pins

They are simply used to input data and display the output data. They can be any number of bits, according to our needs.



8. Logic Gates

AND, OR, NOT and XOR gates were also used.

AND

INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

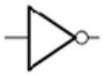
OR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR

INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NOT



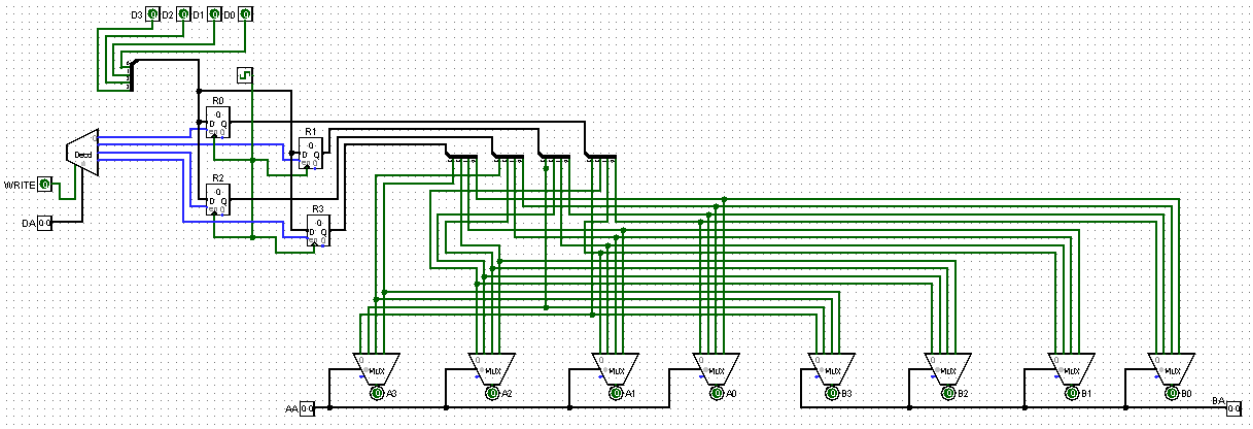
INPUT	OUTPUT
A	
0	1
1	0

Subcircuits

1. 4×4 Register File

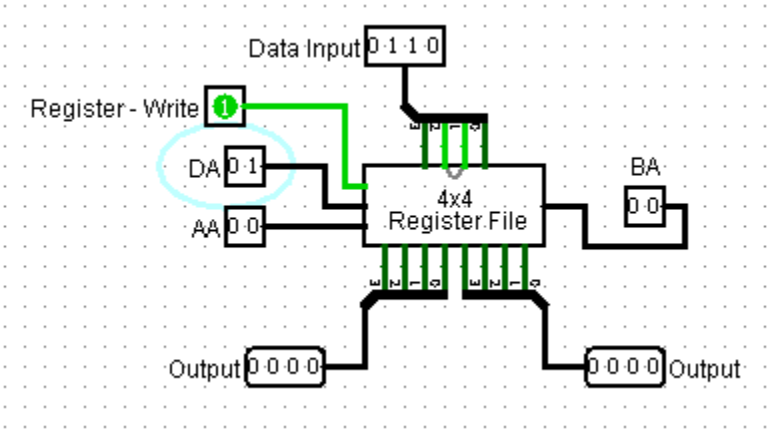
Constructed a 4x4 Register File, which would be able store and transmit 4bit data. A register file is a group of registers, consisting of 2^k registers, needing k bits to access each register using a decoder. The size of the register file is, 2^k x (width of the registers used). In this case, the width is 4 bits and $2^2 = 4$ registers are used, making the size , 4x4. The register file works with the ALU to provide data for processing.

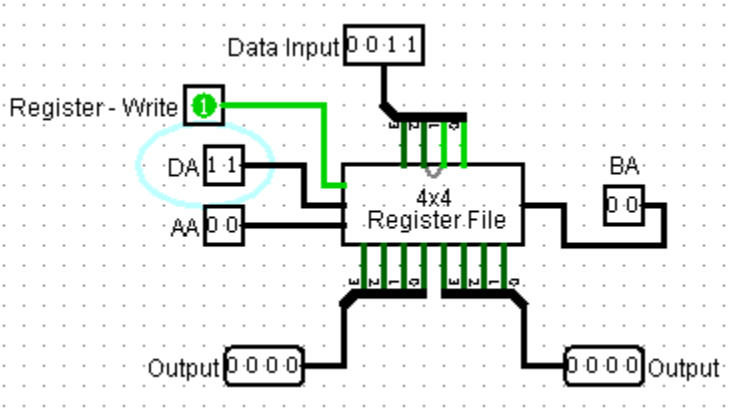
It consists of 4 registers, R0 - R3. A 2x4 Decoder and 8, 4x1 MUXs.



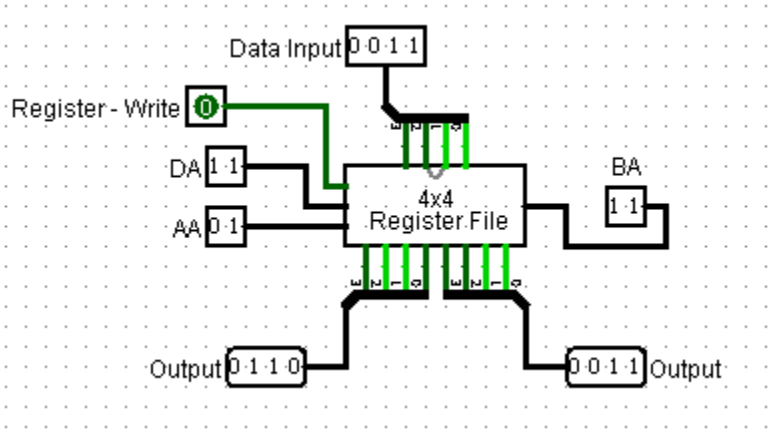
Sample:

- DA controls which register is being written to.
- Data is loaded onto R1(01)





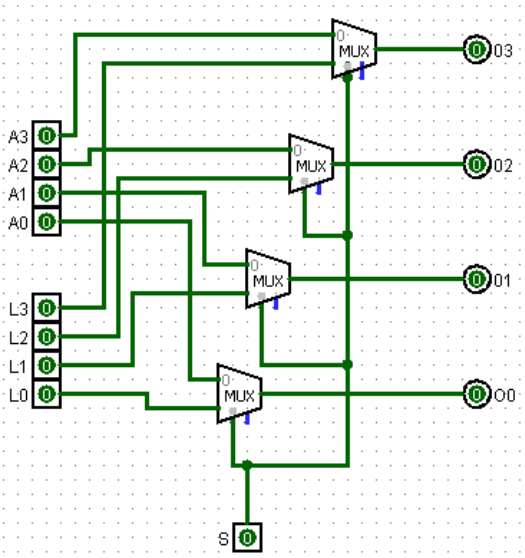
- Data is loaded onto R3(11)



- Data of R1 is displayed at output A (Left output). AA input controls the output of the A output line.
- Data of R3 is displayed at output B (Right output). BA input controls the output of the B output line.

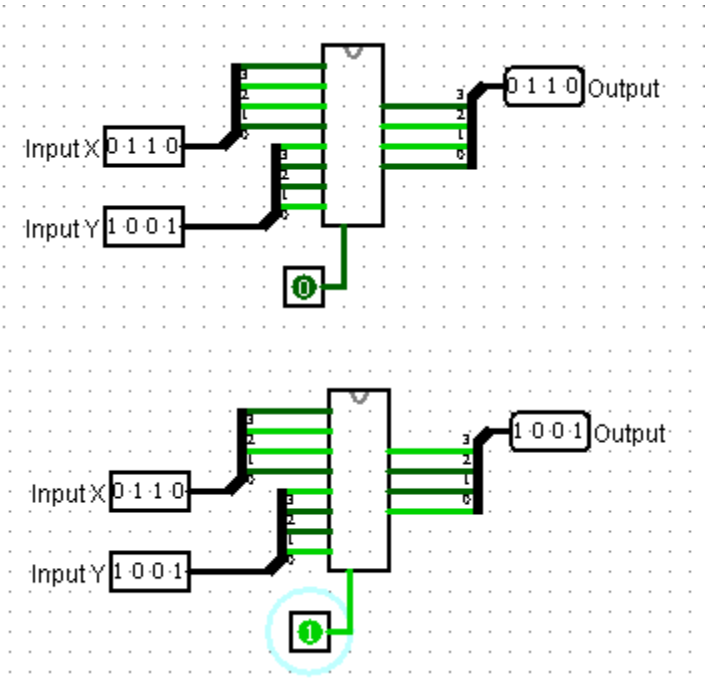
2. Quad set of 2x1 Mux

It acts as a normal Mux would. It accepts 2, 4bit values and outputs one of them, according to the select input. This circuit is used in the ALU and the Data Path, mentioned later.



Sample,

It acts exactly like a normal MUX.

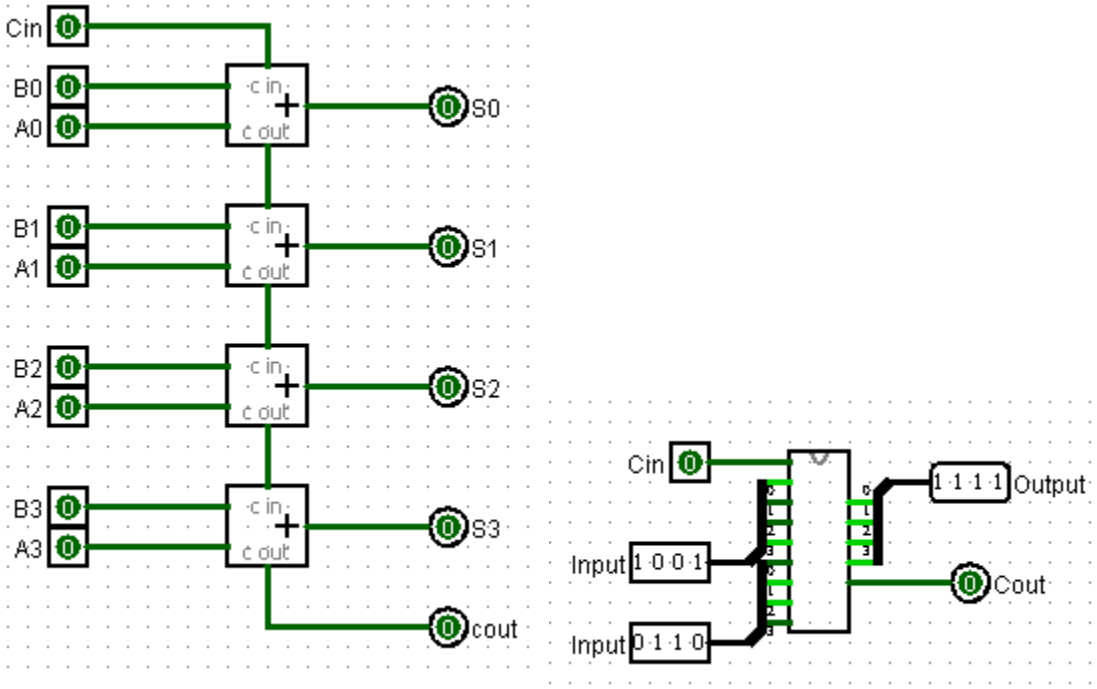


3. ALU – 4bit Operation (Arithmetic Logic Unit)

The ALU consists of an Arithmetic Unit and a Logic Unit. It is used to perform chosen, arithmetic functions or logic functions on 2, 4bit inputs. The functions are controlled through a 4bit control input.

I. Arithmetic Unit

For the arithmetic unit I first constructed a 4bit adder, from 4 1bit adders. The 4bit adder would accept 2, 4bit values and add them. Similar to the 1bit adder.

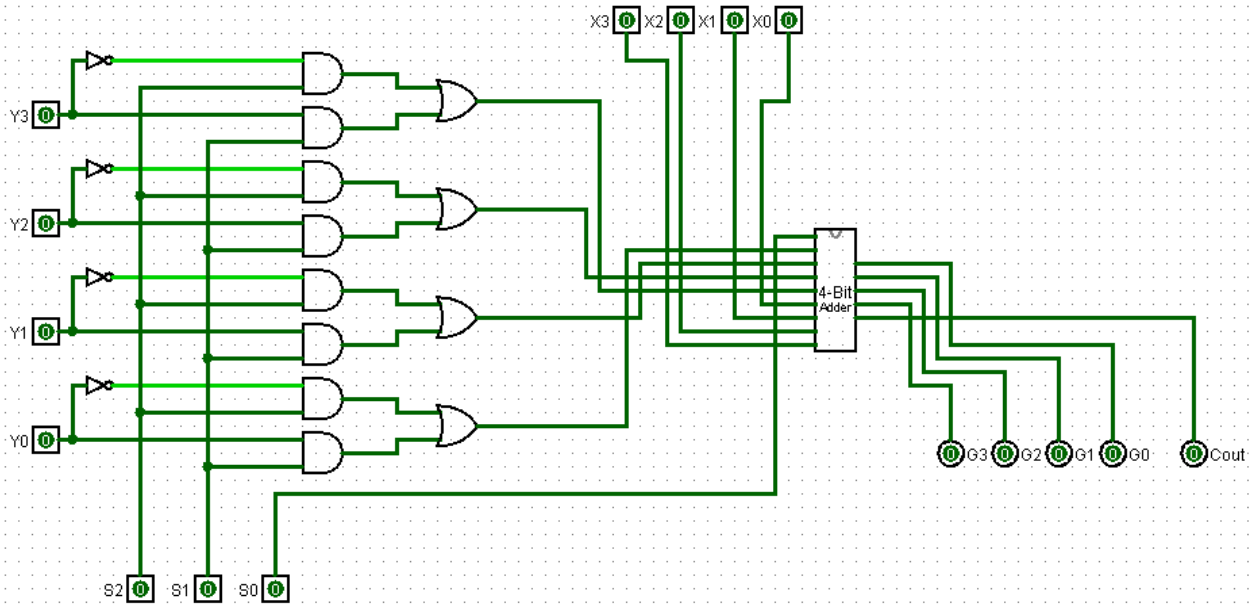


As shown, it simply added the 2, 4bit values.

Then with the 4bit adder and some input logic, consisting of AND, NOT and OR gates, I made the Arithmetic Unit. Which accepts 2, 4bit values and performs operations on them according to the select input, which is 3bits for the time being.

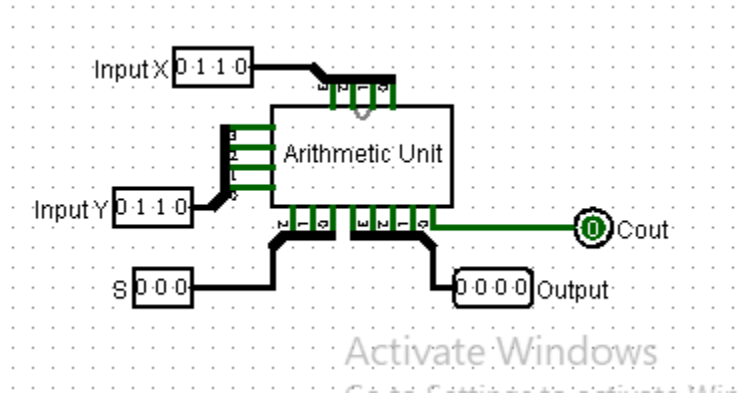
The truth table for the arithmetic unit,

S2	S1	S0	Output	Function
0	0	0	X	Transfer
0	0	1	X+1	Increment
0	1	0	X+Y	Addition
0	1	1	X+Y+1	Addition & increment
1	0	0	X+Y’	1s complement subtraction
1	0	1	X+Y’+1	2s complement subtraction
1	1	0	X-1	Decrement
1	1	1	X	Transfer

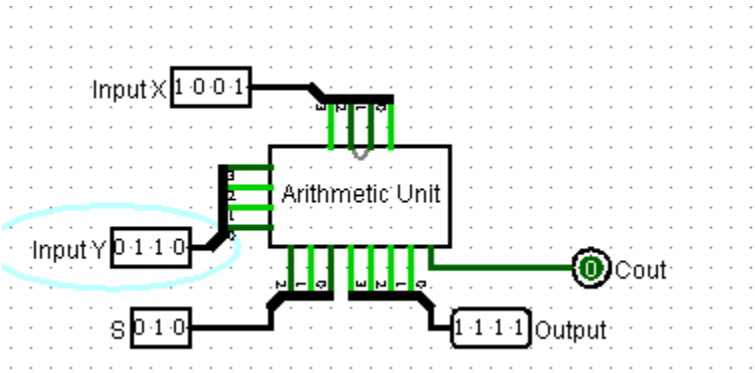


Sample,

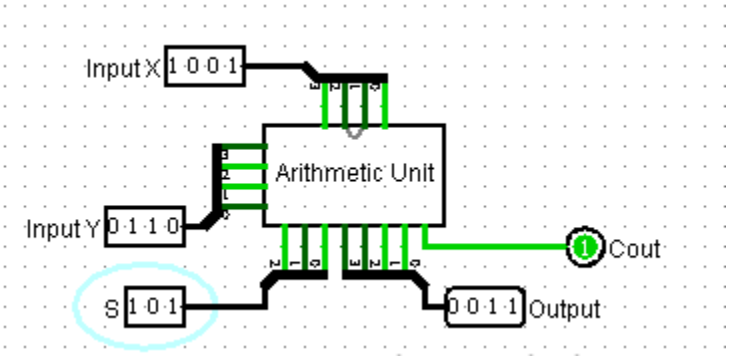
The transfer function being applied, as S = 000.



The Addition function being applied, as S = 010. X+Y = 1001+0110 = 1111



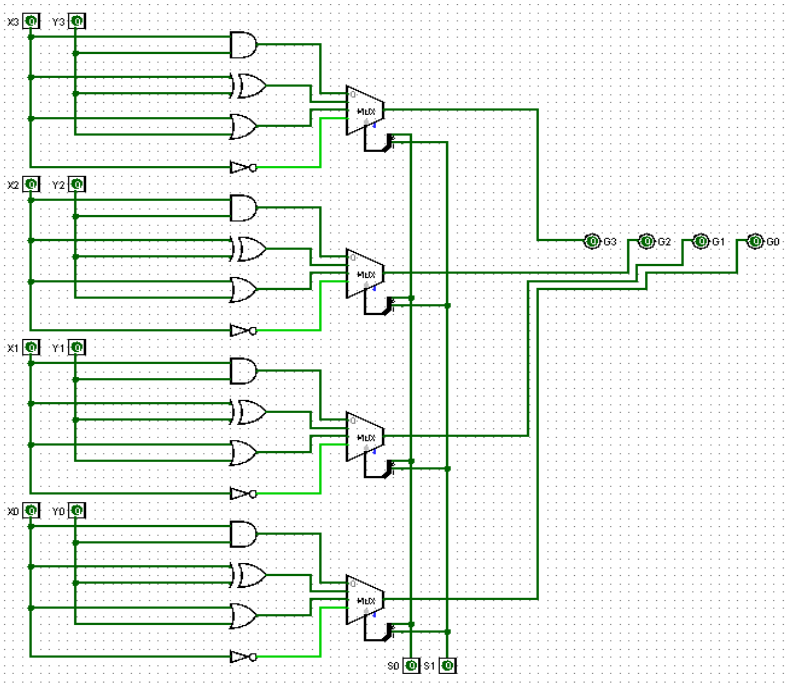
The 2's Complement subtraction being applied, as S = 101. X + Y' + 1 = 1001 + 1001 + 1 = 0011 (Cout = 1)



II. Logic Unit

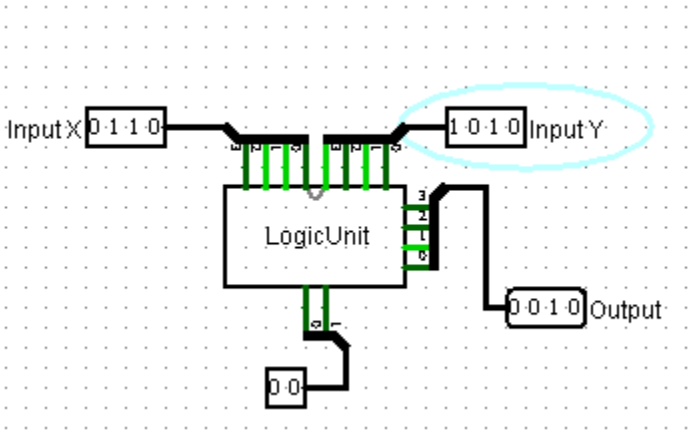
The Logic Unit works similar to the arithmetic unit. It accepts 2, 4bit values and performs a logic operation of our choice, depending on the select input, which are 2bit for now. Its truth table is as follows,

S1	S0	Output
0	0	X AND Y
0	1	X OR Y
1	0	X XOR Y
1	1	X'

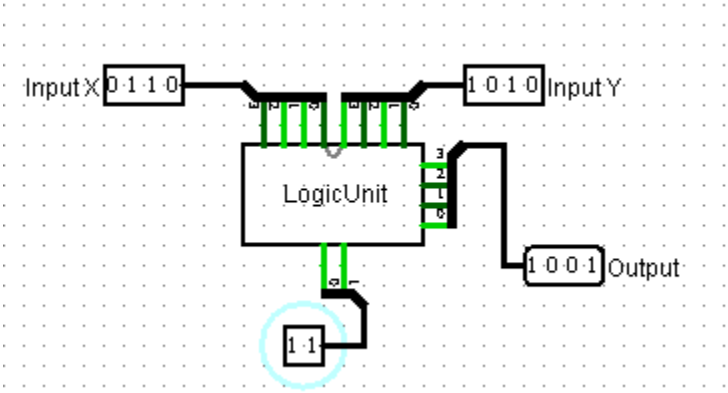


Sample,

S = 00. AND operation being applied to X and Y.



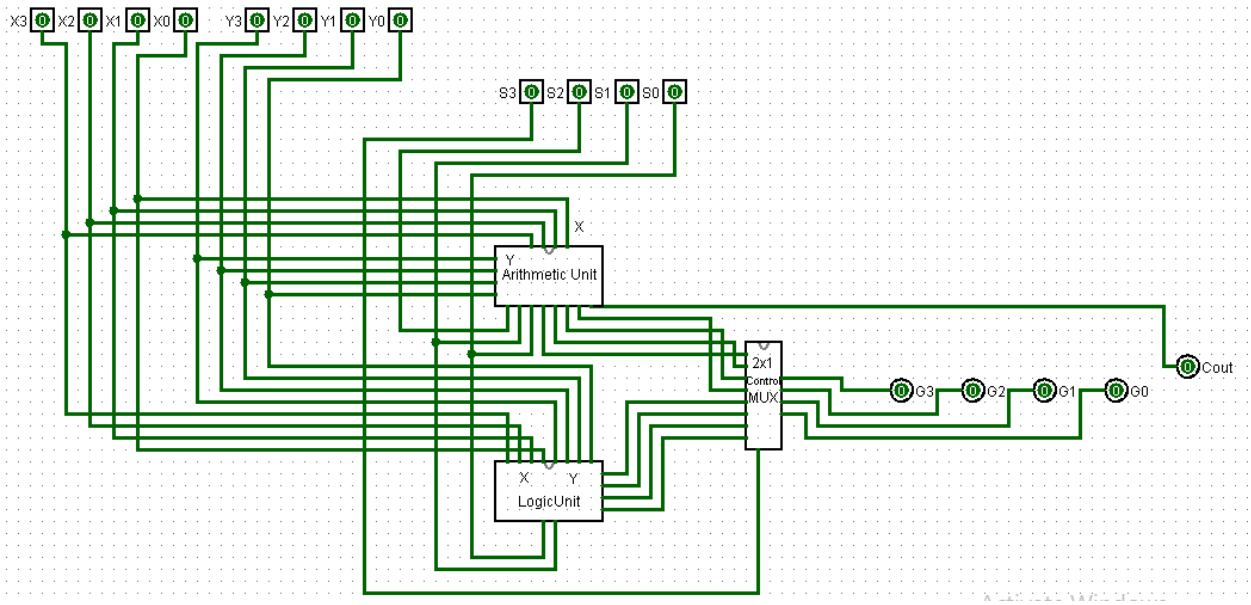
S = 11. Not operation being applied to X



With that we now have both the Arithmetic Unit and the Logic Unit. We combine both to form 1 circuit, the ALU.
The combined truth table of the Arithmetic and Logic Unit, is the truth table for the ALU, as follows.

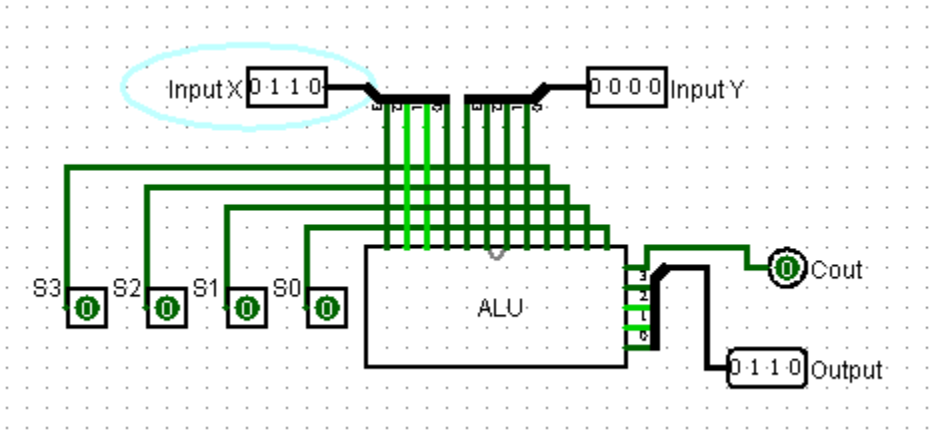
S3	S2	S1	S0	Output	Function
0	0	0	0	X	Transfer
0	0	0	1	X+1	Increment
0	0	1	0	X+Y	Addition
0	0	1	1	X+Y+1	Addition & increment
0	1	0	0	X+Y’	1s complement subtraction
0	1	0	1	X+Y’+1	2s complement subtraction
0	1	1	0	X-1	Decrement
0	1	1	1	X	Transfer
1	X	0	0	X AND Y	AND operation
1	X	0	1	X OR Y	OR operation
1	X	1	0	X XOR Y	XOR operation
1	X	1	1	X’	NOT operation

Alongside the Arithmetic and Logic unit, the Quad set of 2x1 Mux is also used, which was mentioned earlier.

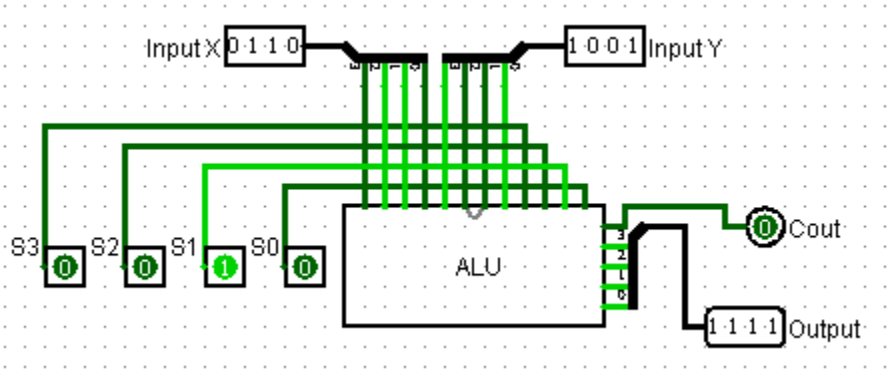


Sample,

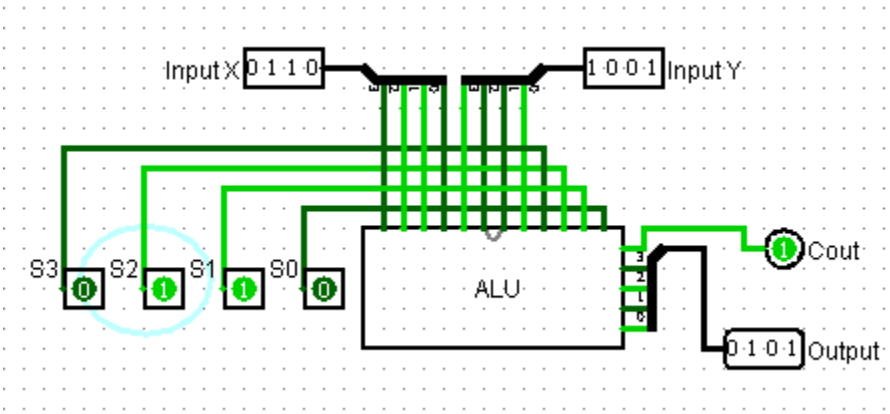
S = 0000. Transfer function is applied. Output = X



S = 0010. Addition is applied. 0110 + 1001 = 1111



S = 0110. Decrement is applied. 0110 – 1 = 0101 (Cout = 1)

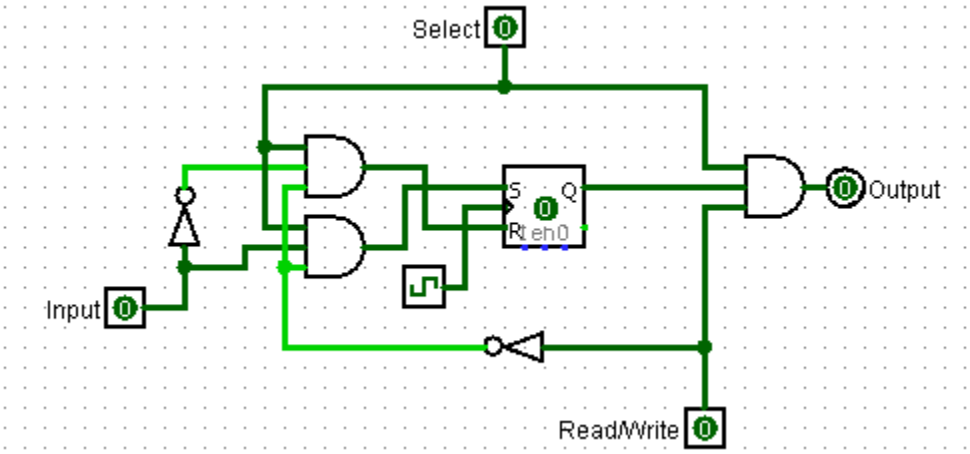


4. 16x4 RAM

This is the main memory of the Data Path. It can be used to store and retrieve data. This 16x4 RAM consists of 16 rows and 4 columns of Binary cells. Containing 16 distinct locations and can operate on 4bit values. A 4bit select input is also required to completely address the 16 locations. A 4x16 decoder is used to access the location we require.

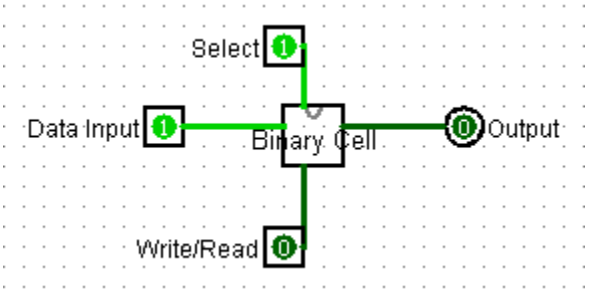
i. Binary Cell:

The binary cell is the building block of the RAM. It can write and read a 1bit value. It is made up of AND gates and a SR flip flop. The select input turns the Binary cell on. The Write/Read controls Writing and Reading. For Writing = 0 and Reading =1.

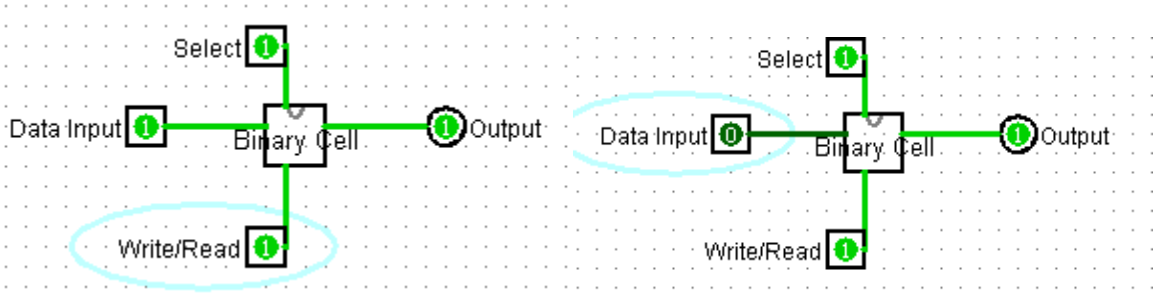


Sample,

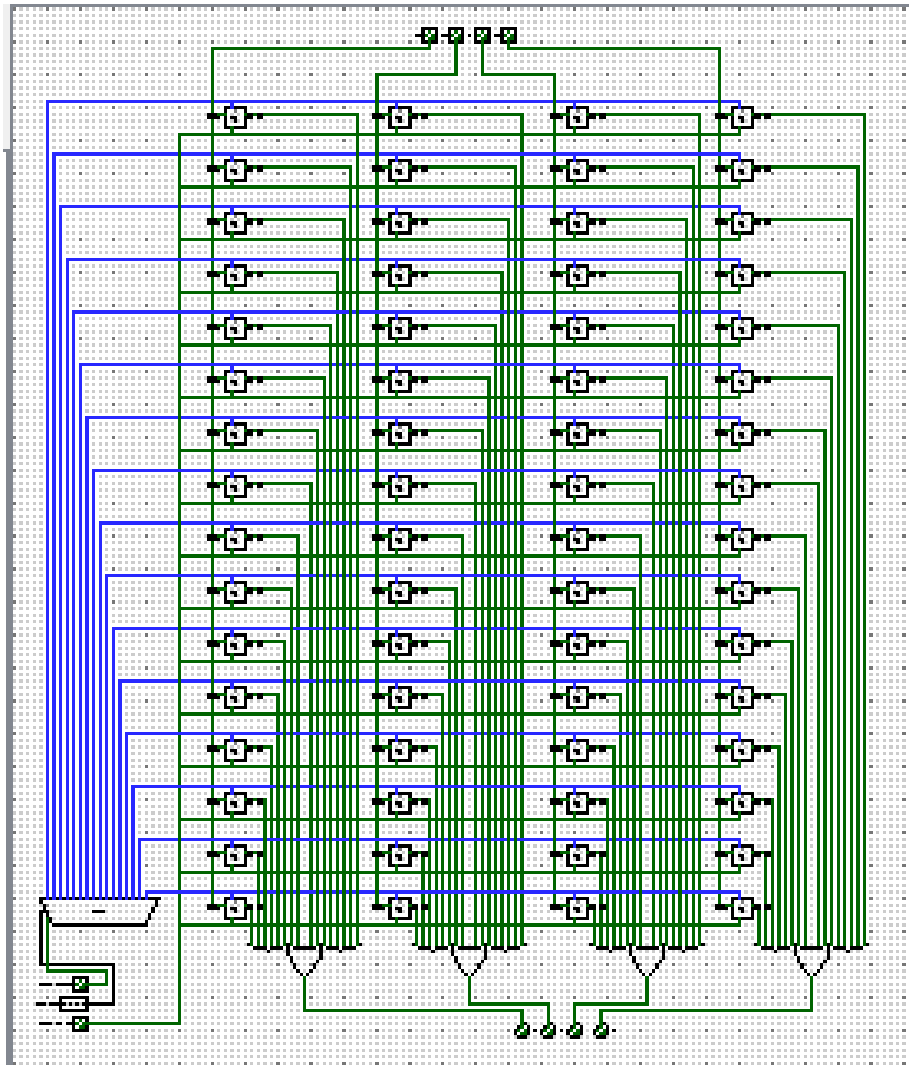
When the select = 1 and Write/Read = 0, the binary cell is writing the data input. With the clock triggers rising edge, the data input is stored in the cell.



When Write/Read = 1, the binary cell is reading, and the previously stored output is displayed. The current output has no effect.

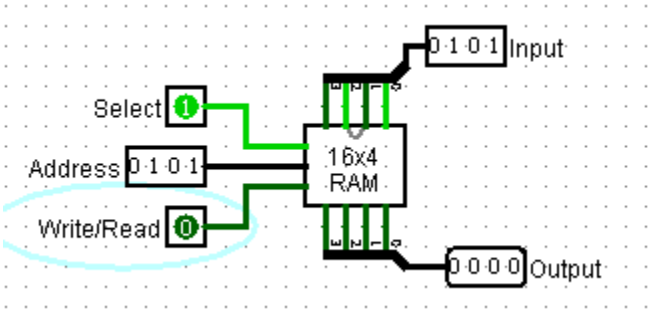


64 Binary cells in total are used to construct this RAM.

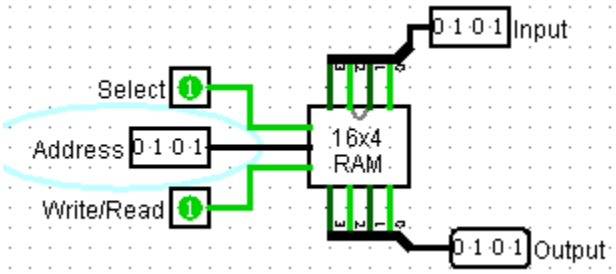


Sample,

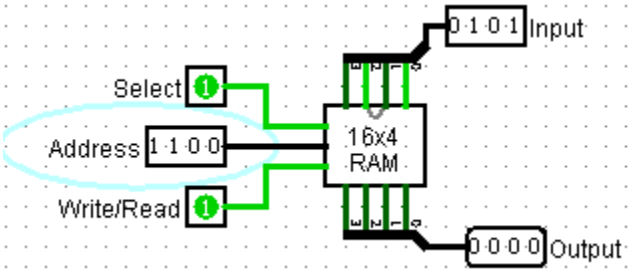
Input data, 0101 is stored in the address 0101.



The data stored at 0101 is displayed. The RAM only outputs the data stored on the address we have selected.



When address is 1100, nothing is output as no value is stored at that address.



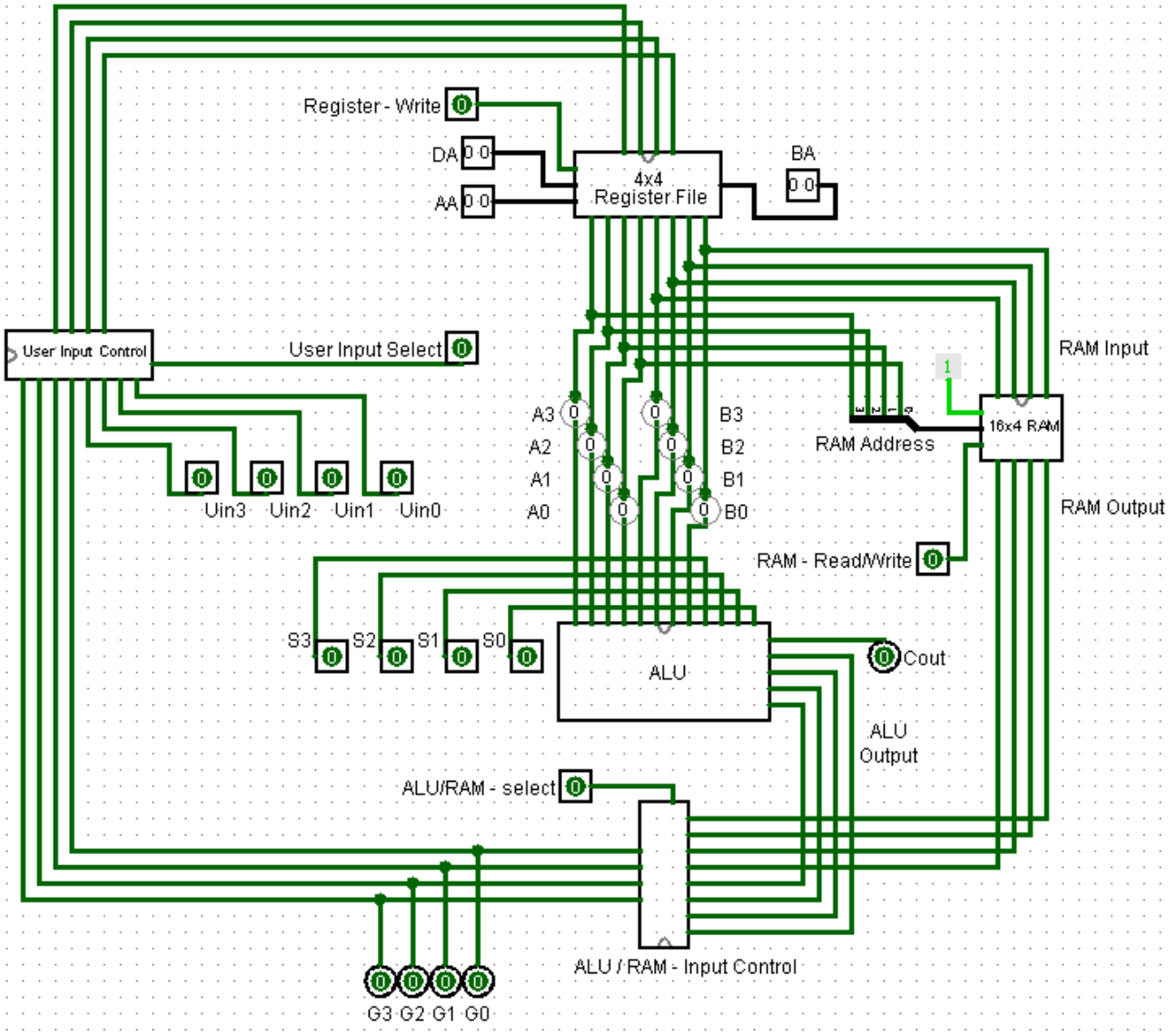
MAIN CIRCUIT

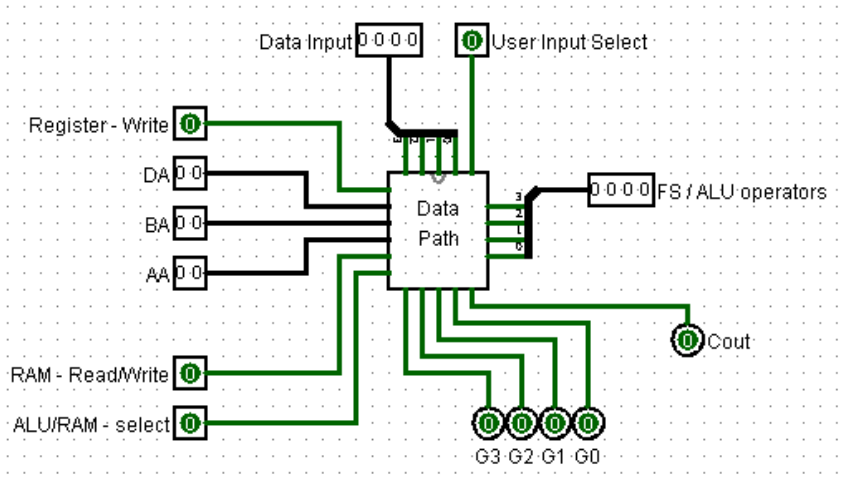
With all the subcircuits mentioned above, we now have all the components necessary to construct the Data Path. The data path uses the register to provide 4 bit values to the ALU for data processing. Between the Register and ALU we connect the RAM, which can store values when we require it to.

We add a Quad 2x1 Mux set at the end, which takes in the ALU and RAMs outputs. We use it to control which of their values is transmitted back to the Register.
Furthermore, we also add another Quad 2x1 Mux set, which controls, whether the Register can receive user input, or the RAM/ALU output, we had chosen earlier.

Using the functionality of each component mentioned earlier, we can transfer data and process it, limited to the operations of the ALU.

Data Path (Circuit & Subcircuit)





LOGIC

- When, User Input Select = 1, user can input his own value to the register file, through Data Input.
  When User Input select = 0, the output the ALU/RAM input control, goes to the register File

- When Register Write = 1, Data can be written onto the Register File.
  When Register Write = 0, Data cannot be written onto the Register File.

- DA selects which register is data written to.
- AA controls which register is accessed to display output at the output A of the register file. (Lines labelled with probes)
- BA controls which register is accessed to display output at the output B of the register file. (Lines labelled with probes)

DA/AA/BA	Register
00	R0
01	R1
10	R2
11	R3

- When, RAM – Read/Write = 0, we can write data to the RAM.
- When, RAM – Read/Write = 1, we can only read data from the RAM.

- When ALU/RAM – select = 0, Data from the RAM is forwarded towards the Register File.
- When ALU/RAM – select = 1, Data from the ALU is forwarded towards the Register File.

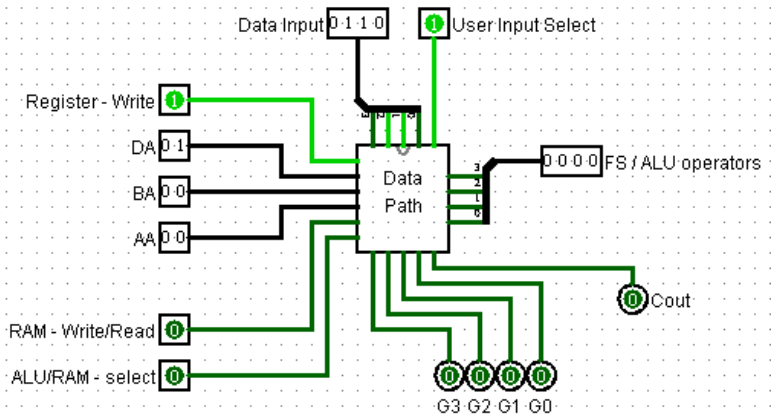
- FS/ ALU Operators, control the operations performed by the ALU. (Operations mentioned in the ALU truth table)

SIMULATION

- Test Run 1

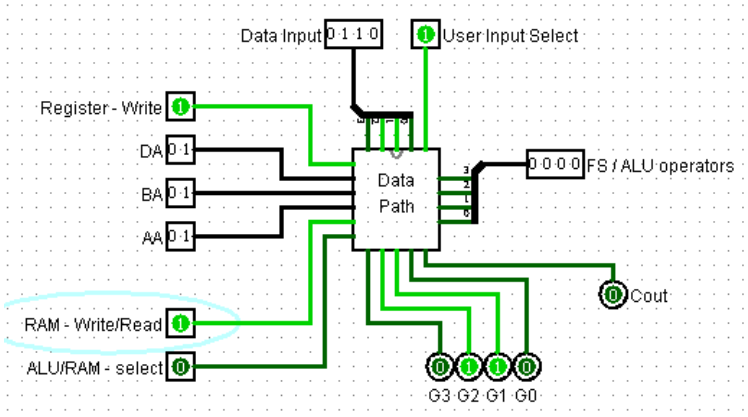
I. R1 ← 0110 (0110 is stored at R1)

User Input Select = 1
Data Input = 0110
Register Write = 1
DA = 01
Clock Pulse



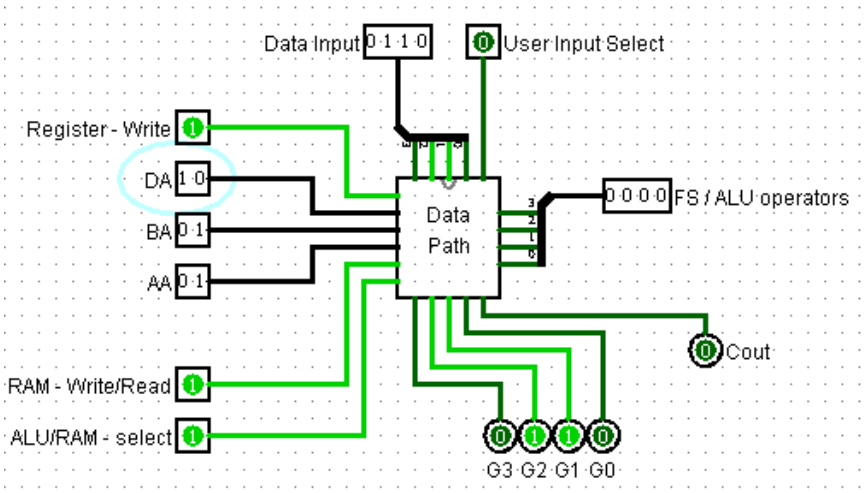
II. M[R1] ← R1 (R1 is stored at 01 in the RAM)

AA = BA = 01
Clock Pulse
RAM Write/Read = 1



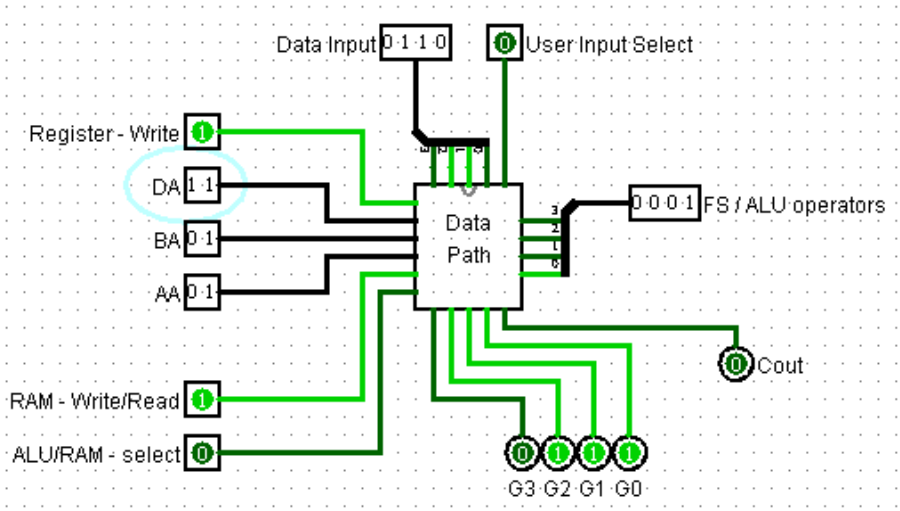
III. R2 ← M[R1] (R2 is written on with M[R1])

ALU/RAM select = 1
User Input Select = 0
DA = 10
Clock Pulse



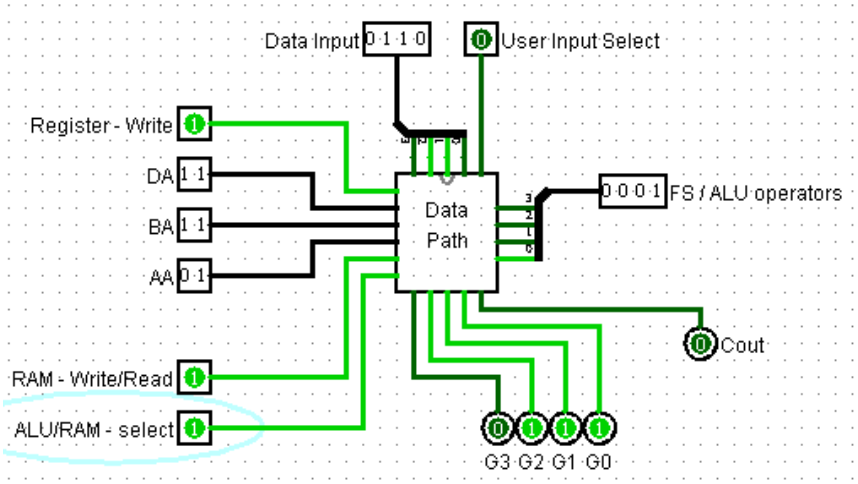
IV. R3 ← R2 + 1 (R3 is written on, with R2 increment)

FS = 0001
ALU/RAM select = 0
DA = 11
Clock Pulse



V. M[R1] ← R3 (01 address in the RAM is overwritten with R3)

RAM Write/Read = 0
AA = 01
BA = 11
Clock Pulse
RAM Write/Read = 1
ALU/RAM select = 1



Results:

M[R1] = 0110

G = 0111

R1 = 0110

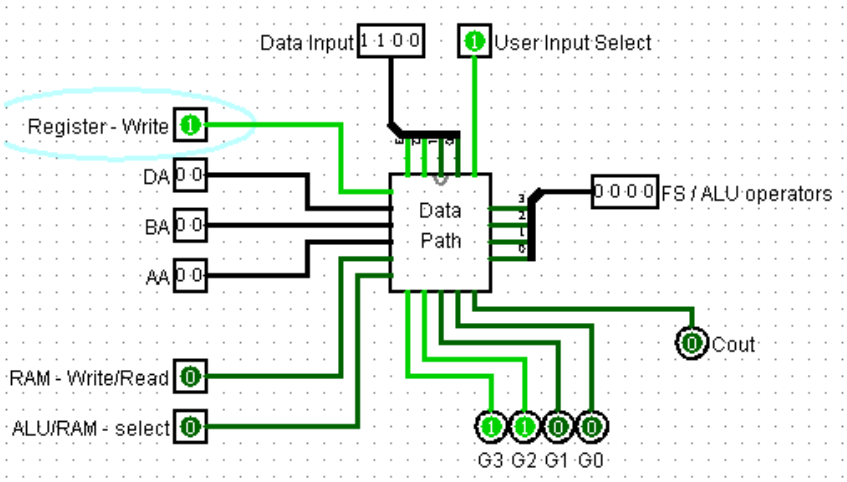
R2 = 0111

R3 = 0110

- Test Run 2

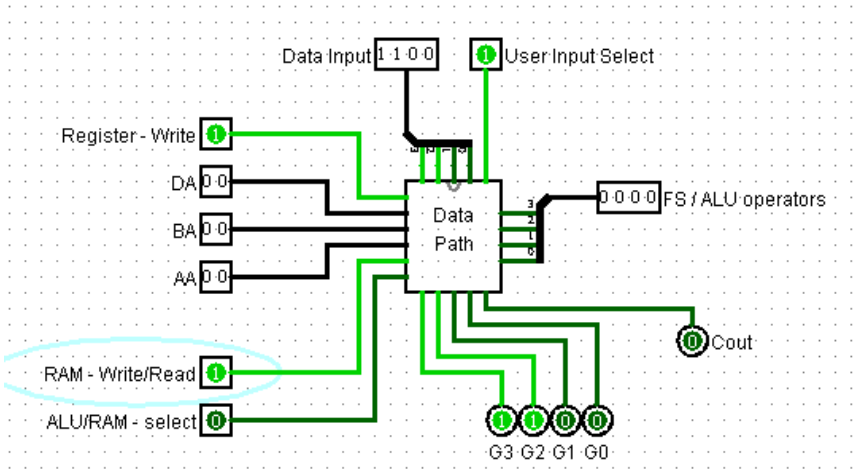
I. R0 ← 1100 (Data is written on register R0)

User Input select = 1
Data Input = 1100
Register Write = 1
DA = 00
Clock Pulse



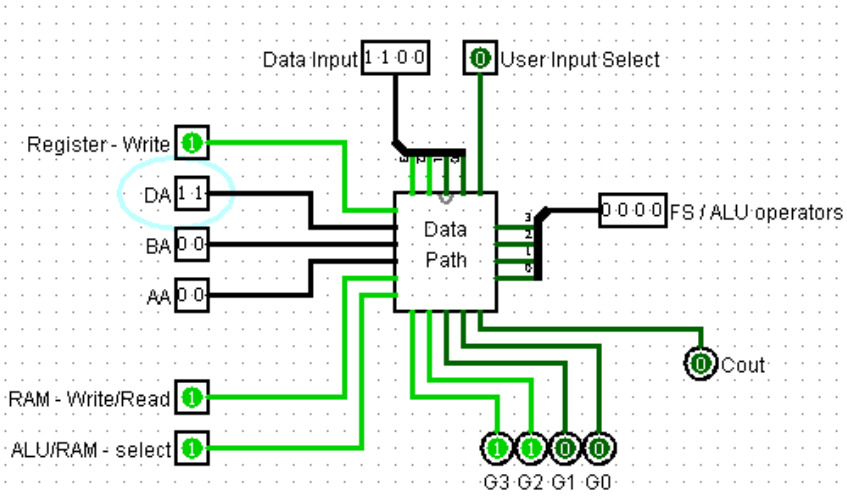
II. M[R0] ← R0 (R0 is stored on 00 in the RAM)

AA = BA = 00
Clock Pulse
RAM Write/Read = 1



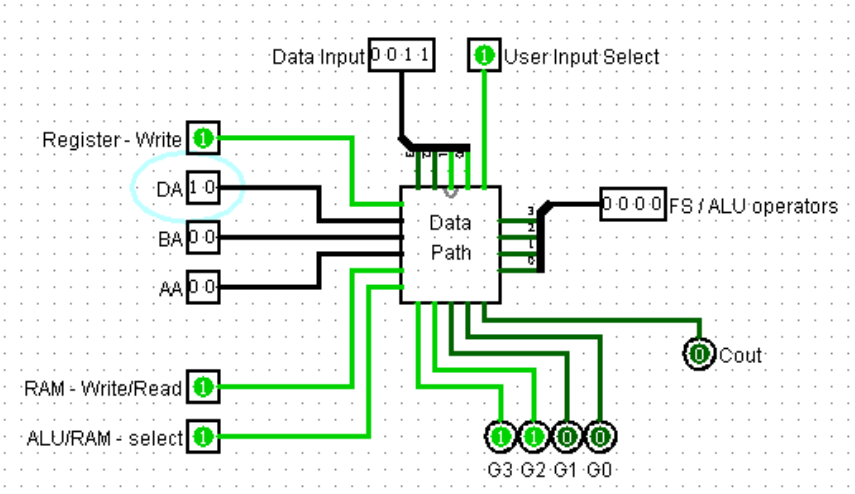
III. R3 ← M[R0] (R3 is written on with the value of M[R0])

ALU/RAM select = 1
User Input Select = 0
DA = 11
Clock Pulse



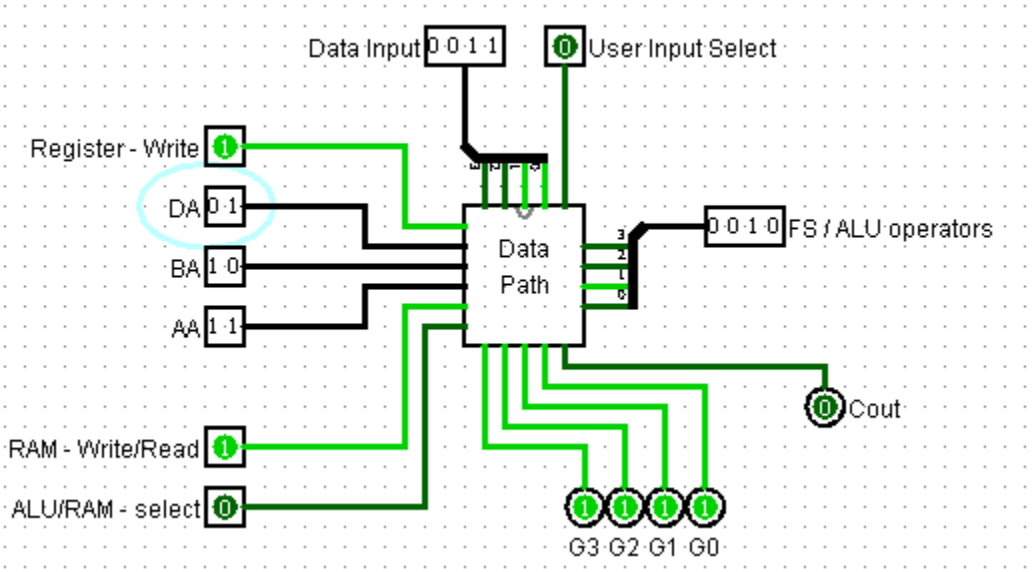
IV. R2 ← 0011 (Data is written on register R2)

User Input Select = 1
Data Input = 0011
DA = 10
Clock Pulse



V. R1 ← R3 + R2

AA = 11
BA = 10
FS = 0010
ALU/RAM select = 0
User Input Select = 0
DA = 01
Clock Pulse



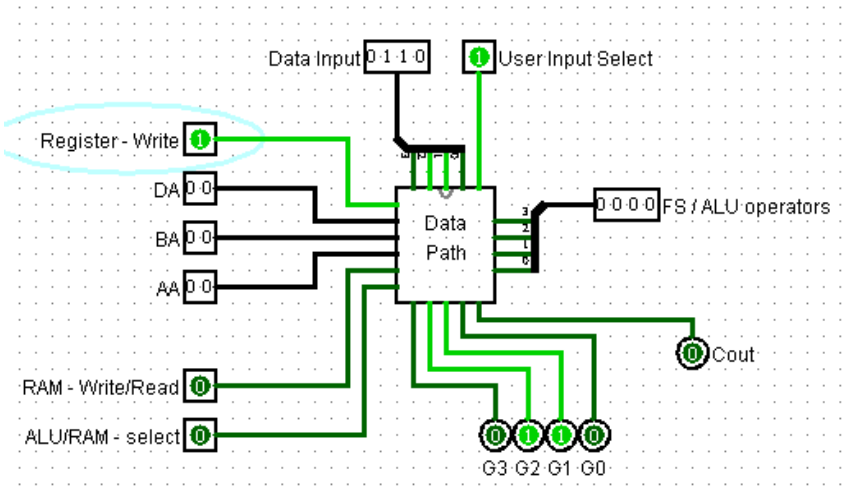
Results:

R0 = 1100
R1 = 1111
R2 = 0011
R3 = 1100
G = 1111
M[R0] = 1100

- Test Run 3

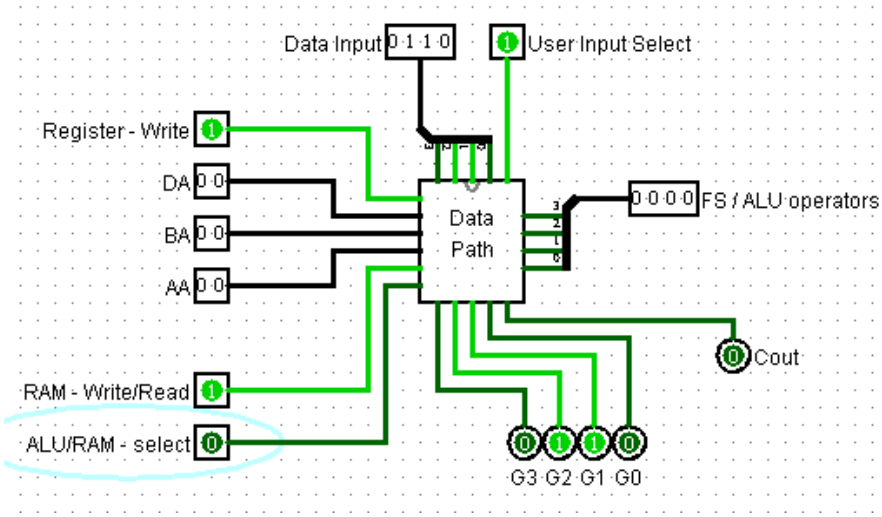
I. R0 ← 0110 (Data is written onto register R0)

User Input Select = 1
Data input = 0110
Register Write = 1
DA = 0
Clock Pulse



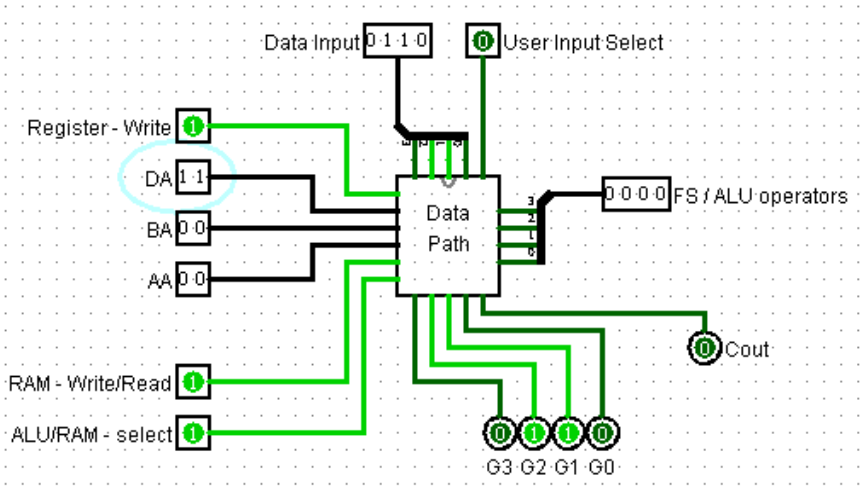
II. M [R0] ← R0 (R0 is stored on 00 in the RAM)

AA = BA = 00
Clock Pulse
RAM Write/Read = 1



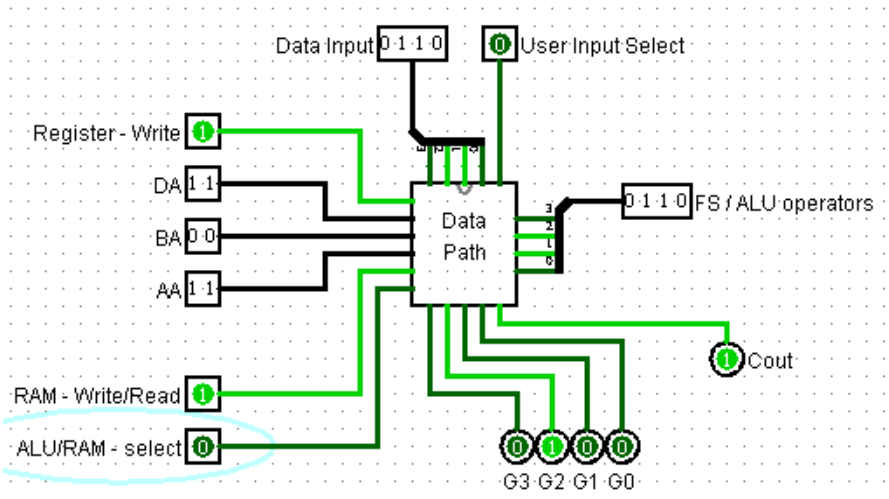
III. R3 ← M [R0] (Register R3 is written on by the value stored at 00 in RAM)

ALU/RAM select = 1
User input select = 0
DA = 11
Clock Pulse

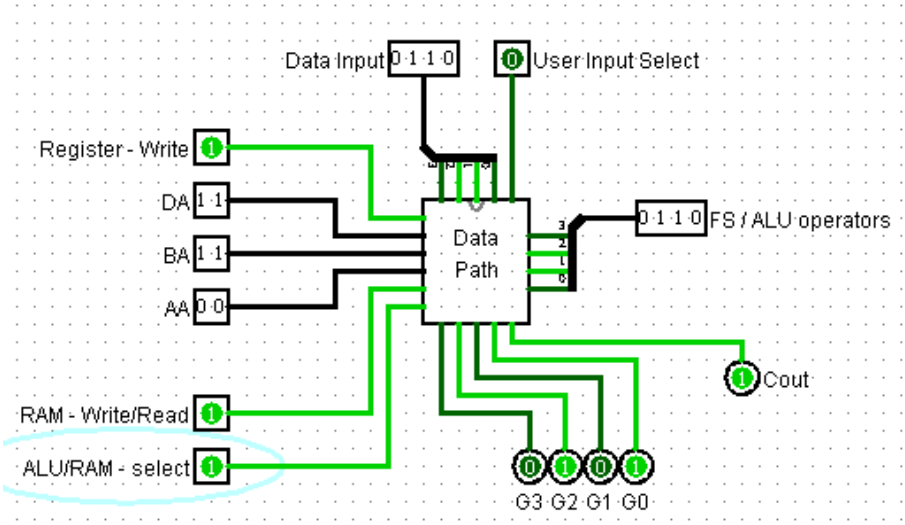


IV. R3 ← R3 – 1 (R3 is decremented and is overwritten)

AA = 11
FS = 0110
ALU/RAM select = 0
DA = 11



V. M [R0] ← R3 (M [R0] is overwritten with the value from R3)
AA = 00
BA = 11
RAM Write/Read = 0
Clock Pulse
RAM Write/Read = 1
ALU/RAM select = 1



Results,

R0 = 0110
R3 = 0101
M [R0] = 0101
G = 0101