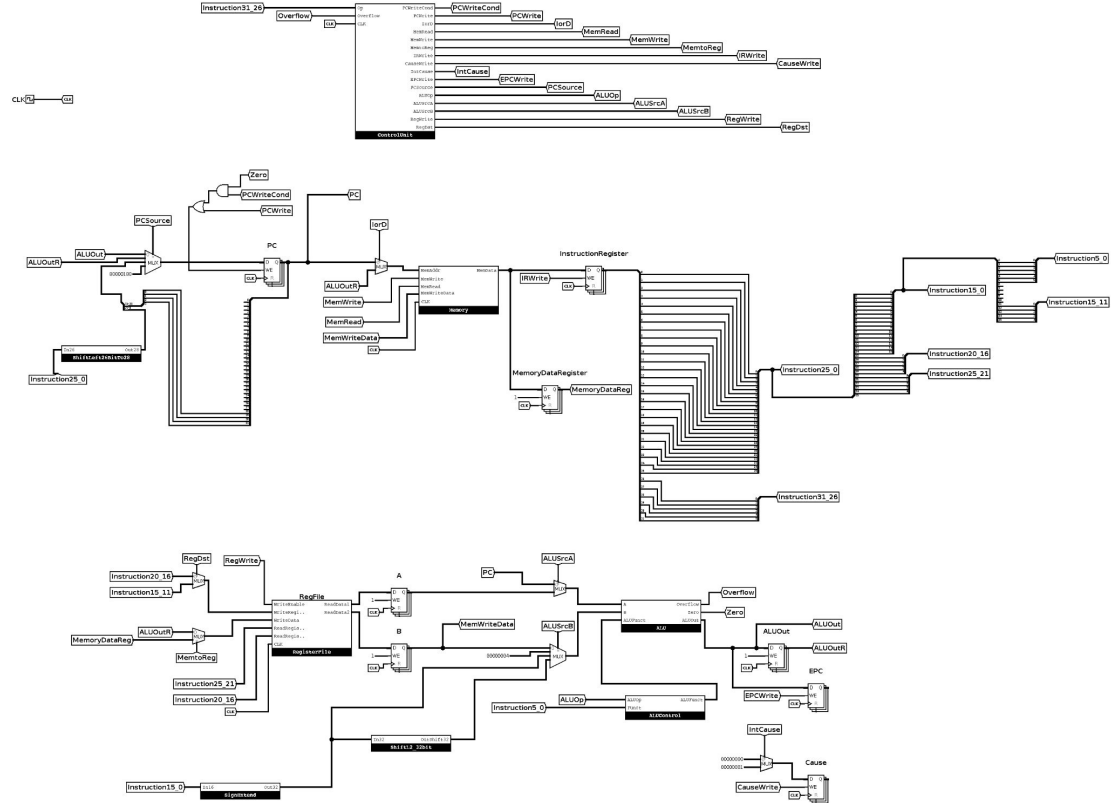


# Multicycle datapath in Logisim

# General structure

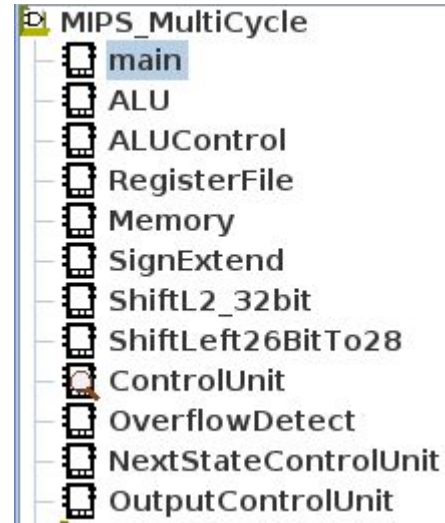


# Sub-components

You can “main” circuit is made of various sub-components.

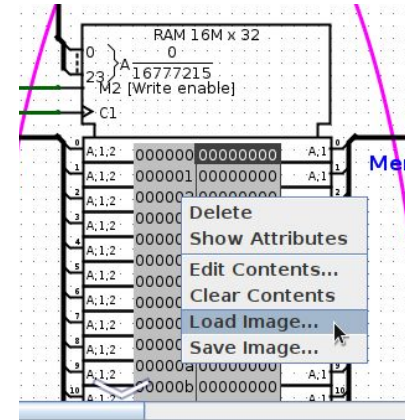
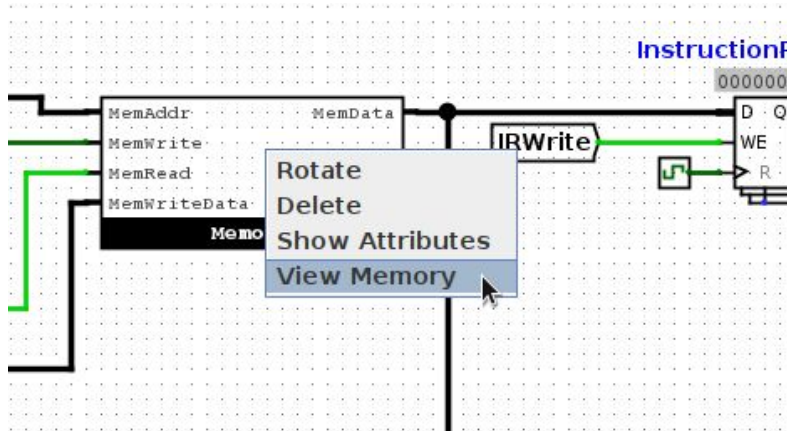
You see the various components of the datapath using the menu on the left.

If you click on them you can see the circuit.



# Load a program (1)

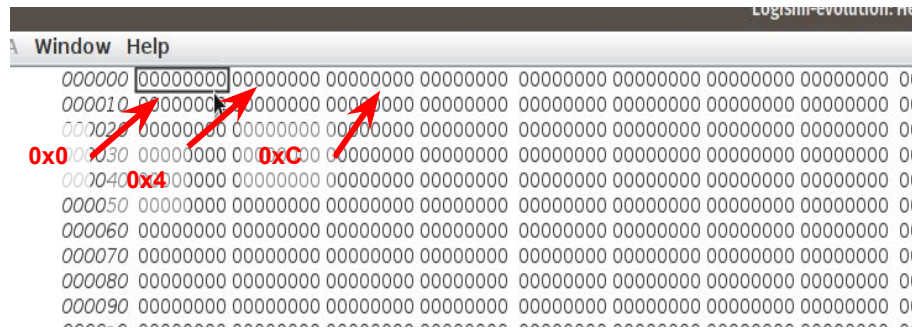
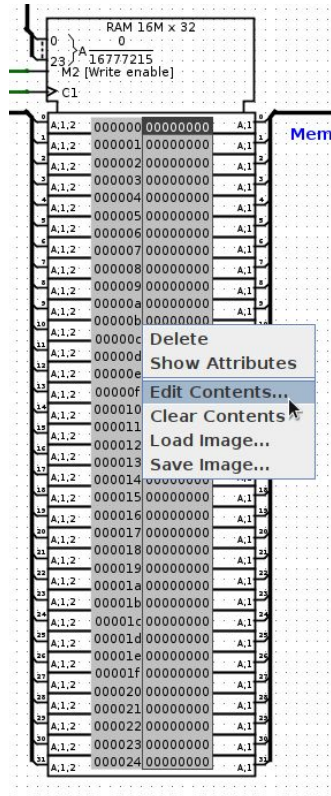
Go in the “main”, right click on “Memory” -> “View Memory”  
Right click on RAM -> “Load image...” and select the .hex file to store in the memory



## Load a program (2)

You can also manually insert instructions and data in hexadecimal.

WARNING: the addresses in the memory are word aligned, you cannot address by bytes

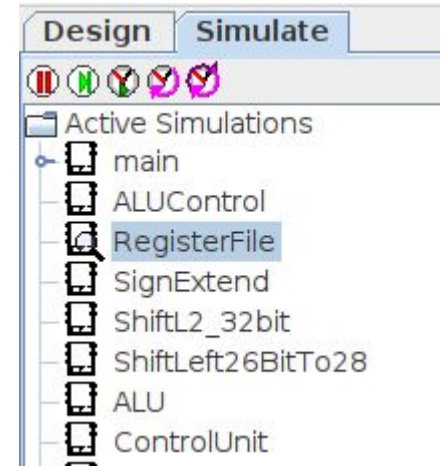


# Execute a program

Verify that “Run simulation” is ticked.

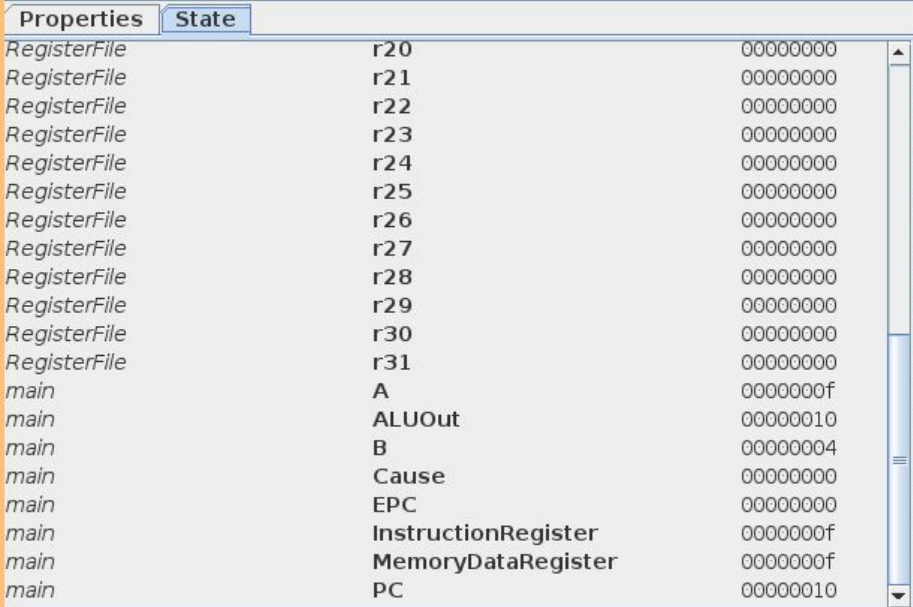
Click on “Tick Full Cycle” (or press F9) to do one clock cycle.

You can verify the datapath steps by clicking on the components in the “simulate” menu



# Watching registers values

You can watch the current values of the registers in the “state” tab, in the bottom left corner of logisim.



The screenshot shows the 'State' tab of the Logisim interface. It displays a list of components and their current values in hexadecimal. The components include 32 registers (r20-r31), ALUOut, B, Cause, EPC, InstructionRegister, MemoryDataRegister, and PC. The first 12 registers (r20-r31) are of type 'RegisterFile' and all have a value of 00000000. The next 5 components (A, ALUOut, B, Cause, EPC) are of type 'main' and have values 0000000f, 00000010, 00000004, 00000000, and 00000000 respectively. The InstructionRegister, MemoryDataRegister, and PC are also of type 'main' and have values 0000000f, 0000000f, and 00000010 respectively. A scrollbar is visible on the right side of the table.

Properties	State	
RegisterFile	r20	00000000
RegisterFile	r21	00000000
RegisterFile	r22	00000000
RegisterFile	r23	00000000
RegisterFile	r24	00000000
RegisterFile	r25	00000000
RegisterFile	r26	00000000
RegisterFile	r27	00000000
RegisterFile	r28	00000000
RegisterFile	r29	00000000
RegisterFile	r30	00000000
RegisterFile	r31	00000000
main	A	0000000f
main	ALUOut	00000010
main	B	00000004
main	Cause	00000000
main	EPC	00000000
main	InstructionRegister	0000000f
main	MemoryDataRegister	0000000f
main	PC	00000010

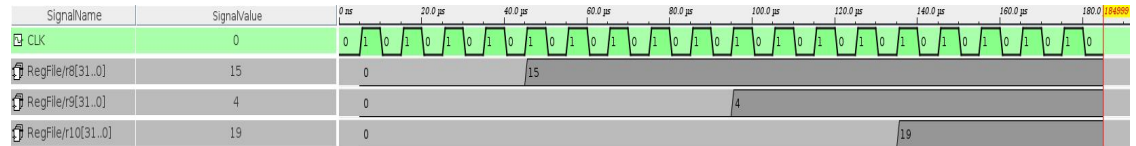
# Timing diagrams

You can also watch the evolution of the various signals and registers clock cycle by clock cycle.

In the “simulate” menu click on “Timing Diagram”

Select CLK as the clock source, then select what signals you want to watch.

After this steps you can tick the clock and watch the signals evolve





# Tips & Tricks for modifying the datapath

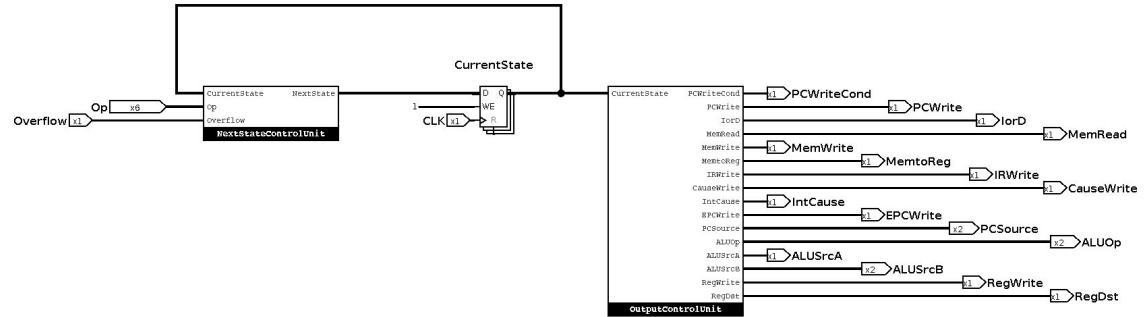
# Control Unit (1)

The control unit is made from 2 combinational circuits and a register.

The first combinational block outputs the next state based on the current state, the opcode and the overflow signal.

The next state gets written to the StateRegister on every rising edge.

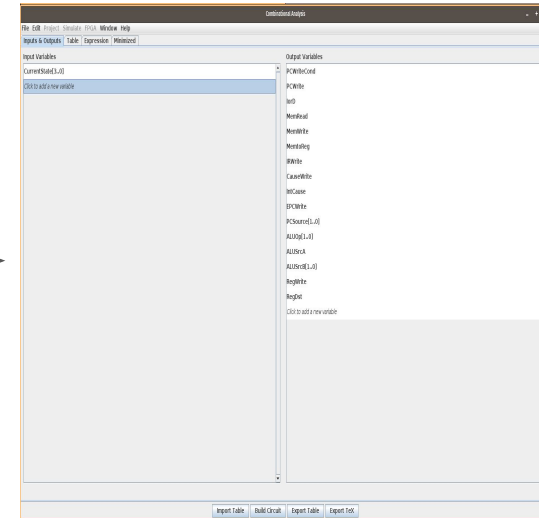
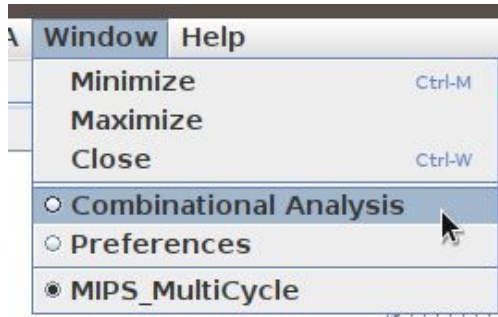
The output of the StateRegister is also used by the second combinational block to decide the outputs signals.



# Control Unit (2)

The 2 combinational blocks are generated by a Logisim Evolution tool, that allows us to create a circuit based on its truth table

You can import truth tables from the .txt files in the repo (the same is true for the ALU Control)



# Control Unit (3)

In the “Table” tab you can see the truth table of the circuits.

The bit indicated by “-” are “don’t cares”.

CurrentState[3..0]	Op[5..0]	Overflow
0 0 0 0	- - - - -	-

## Control Unit (4)

You have to modify both the “NextStateControlUnit” and the “OutputControlUnit” truth tables and generate the new circuits.

Import the .txt, modify the truth table as needed and then click “Build Circuit”. Enter the same name and overwrite the previous circuit. You can also save the modified truth table by clicking on “Export Table”.

# Modifying the width of a signal (1)

## Modifying the width of a signal (2)

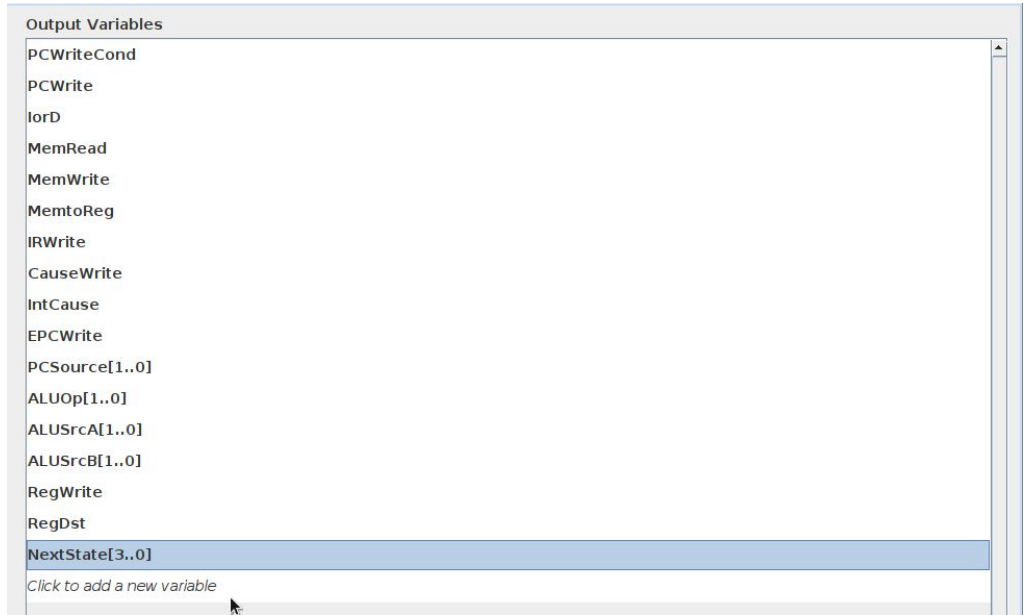
When you modify the width of the signals you also need to modify the components that use the signals.

Selection: Multiplexer	
VHDL	Verilog
Facing	East
Gate Size	Wide
Select Location	Top/Right
Select Bits	2 ▼
Data Bits	32
Disabled Output	Zero
Include Enable?	No

Selection: Tunnel "ALUSrcA"	
VHDL	Verilog
Facing	South
Data Bits	2 ▼
Label	ALUSrcA
Label Font	SansSerif Bold 16

# Add control signals (1)

To add a new control signals go in “Inputs & Outputs”, click “Add a new variable” , write the name of the signal and select the number of bits.





## Add control signals (2)

After building the new circuits you can find the signal in “main”.

Add a “Tunnel” to use the signal inside the datapath.



You can find everything on my github:

<https://github.com/fdila/MIPS-multicycle-datapath>

Open a issue if you find something wrong, and Pull Requests are also welcome!