| Computer Architecture & Assembly Language | | |
|---|---|---|
| Faisal Iradat, PhD<br>Muhammad Zain, PhD Scholar | Lab – 9,10 and 11<br>(Exploring RISC-V 32bit Single Cycle CPU Implementation in Logisim Evolution) | Marks: [3+2+5 = 10]/10*3 |
| Reference:<br>• UC Berkeley<br>• T-K Github Public Repository | | |

In this lab, you will:

**Objectives:**

- Explore an existing simple design and implementation of RISC-V Single Cycle CPU

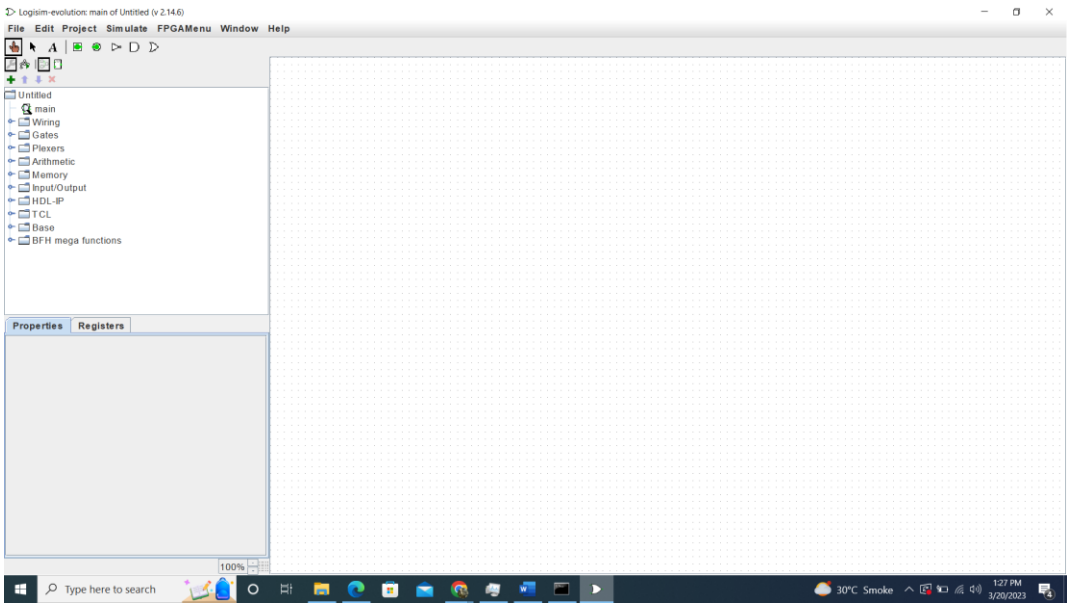
**Logisim Evolution**

In the previous lab, Logisim was introduced, and you were asked to implement the Program Counter and Register File. Designing circuit in Logisim is exciting and fun. Real designs can be simulated in Logisim and verified before chip design. In this lab you will be using Logisim Evolution to build RISC-V CPU. Logisim Evolution is no different from Logisim, supports complex designs and available for all OS platforms.

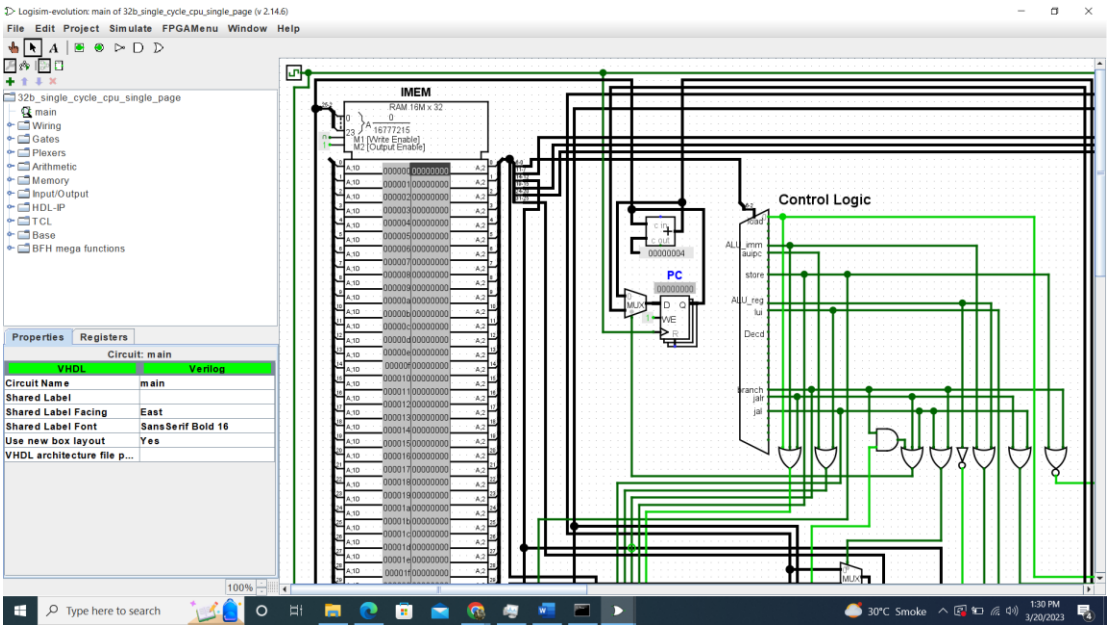
**RSIC-V 32-bit Single Cycle CPU**

We will use an existing implementation of RISC-V 32-bit single cycle CPU in Logisim developed at UC Berkeley. In this implementation, the Code loads different memory values and print them on a LED matrix. Follow the steps below to understand and simulate the processor.

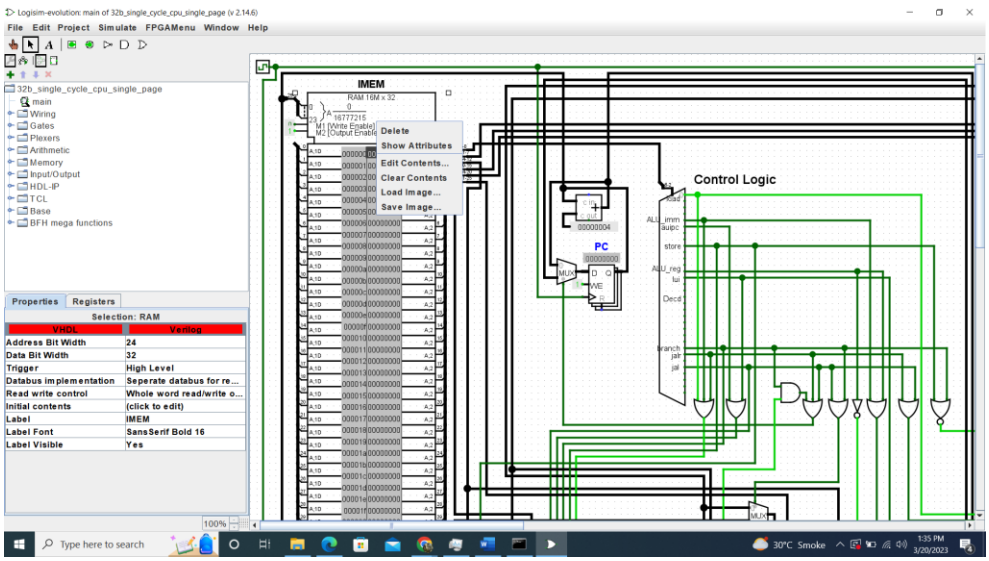
1. Go to your LMS account and download 'RISC-V-Single-Cycle-CPU-master.zip' file.
2. Extract the file in your preferred directory.
3. After extraction, go in the extracted folder and execute 'logisim-evolution.jar' file. You should have Java installed in your machine. The following screen will be displayed.



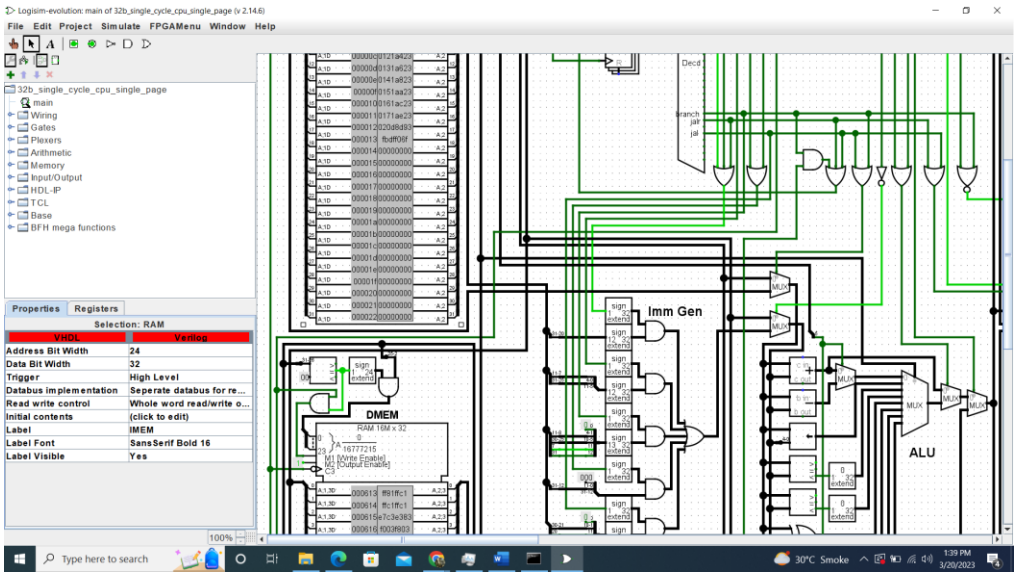

4. Open the '32b_single_cycle_cpu_single_page.circ' file in logisim-evolution.



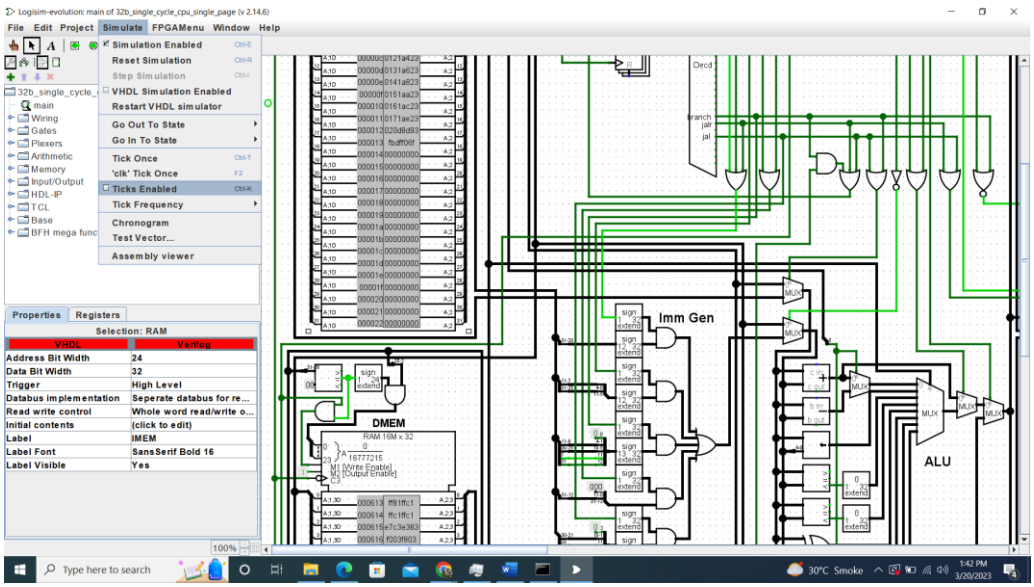

5. Right click the IMEM RAM and click the 'Load Image' option. Load the IMEM file from your extracted folder.



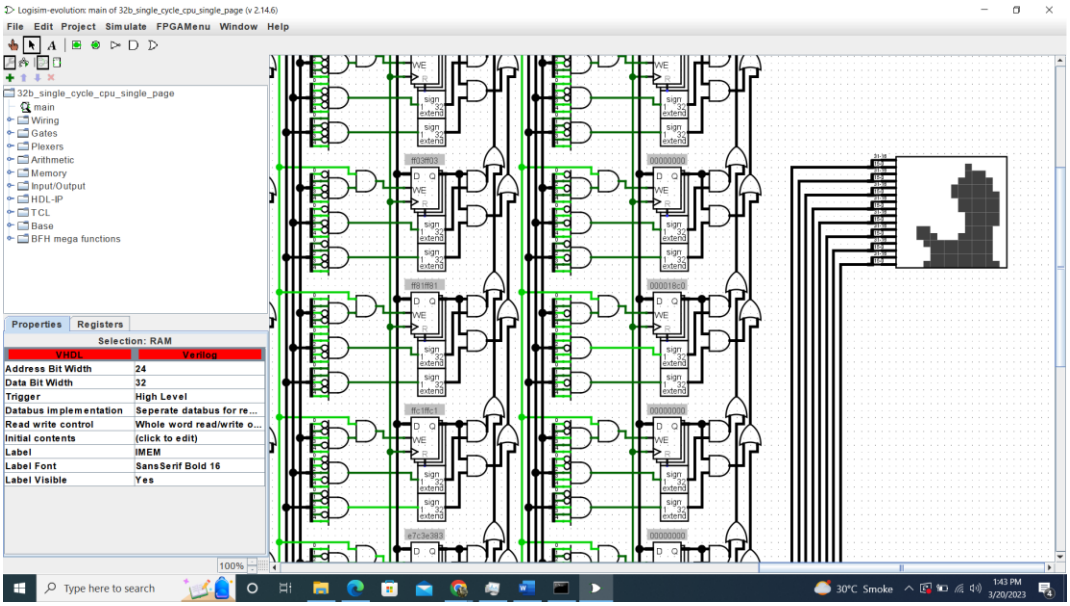

6. Repeat step 5 for DMEM and load the DMEM file.



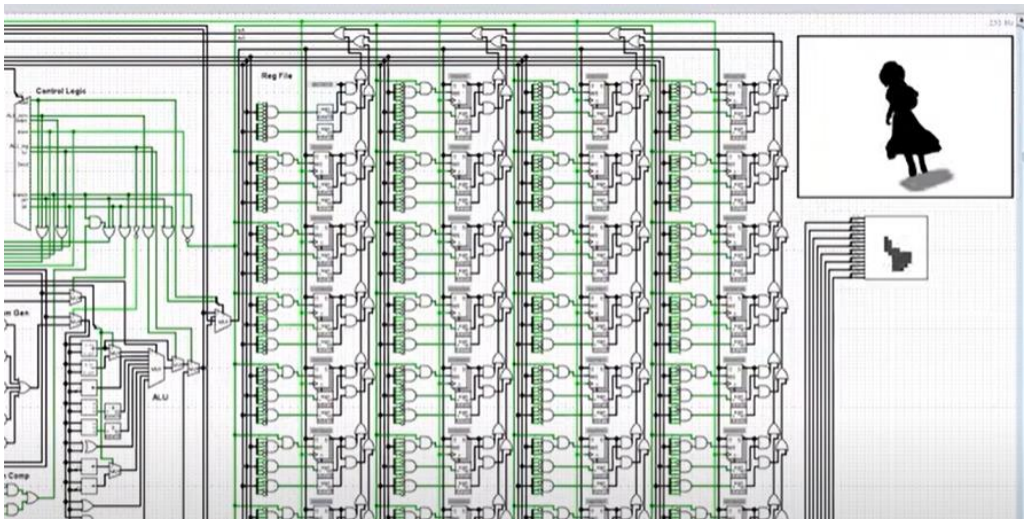

7. After loading the IMEM and DMEM files go to ‘Simulate’ menu, change the frequency to 512Hz and select the ‘Ticks Enabled’ option.




8. Now view the LED matrix and you will see some figure is drawn.




The simulation approximates different real images an example is shown below:




**How does this work?**

The program ‘testing.s’ in your extracted folder contains the code. Each line of code in converted into machine code and written in the IMEM file. You can verify by converting each line in the program file to machine code and check it in the IMEM file. The IMEM and DMEM files are both editable.

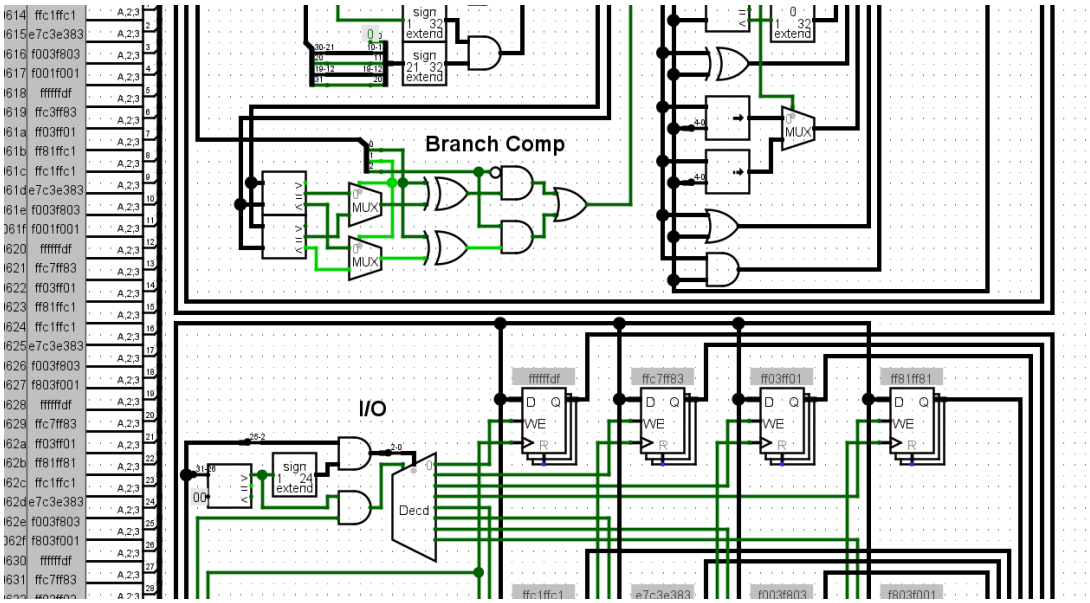
The entries in the DMEM file was created using the python script ‘convert.py’ for different images. For fun you can do it at your own, however, not a requirement for this lab.

**Exercise**

Create your public GIthub Repository to upload the solution of exercises below.

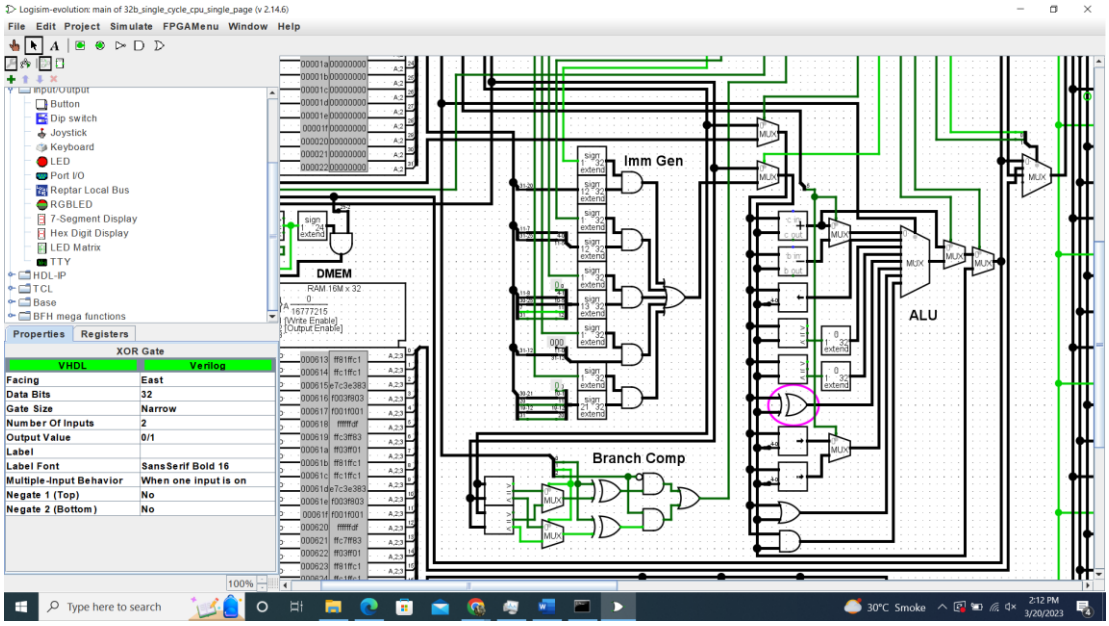
Q1. Do the following:

a. Write a program that stores your name and id in the memory. Convert each line into machine code and paste it in the IMEM file.
b. Now change the circuit and use TTY (please refer to Logisim documentation or use of TTY from the web) in place of the LED matrix. You will have to make several changes in the I/O section of the implementation.




c. Display your name and id from memory on TTY.

Q2. Redesign the CPU by converting the AND, OR and XOR gates using NAND gates.




Q3. Down scale the existing implementation to 16-bit