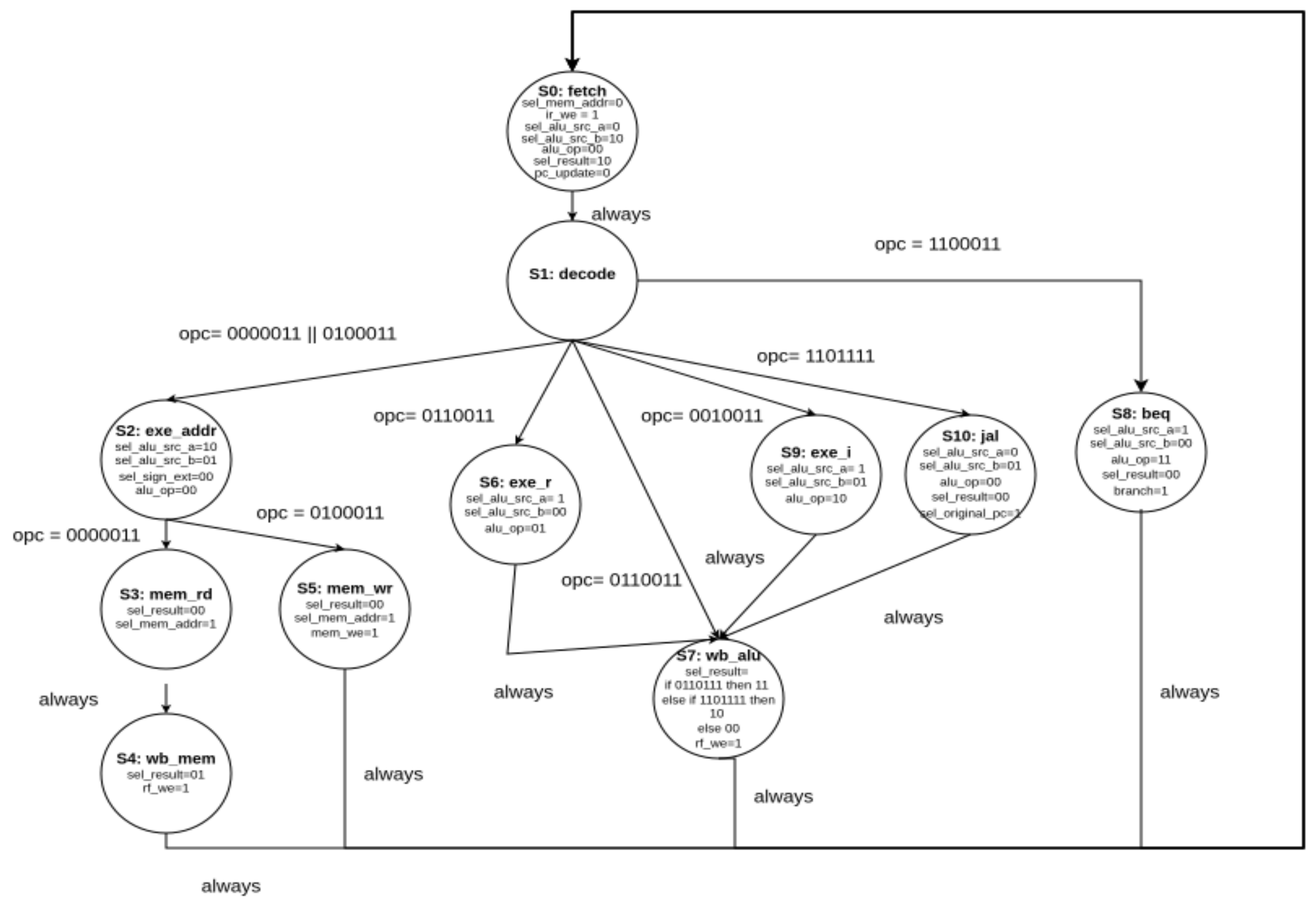
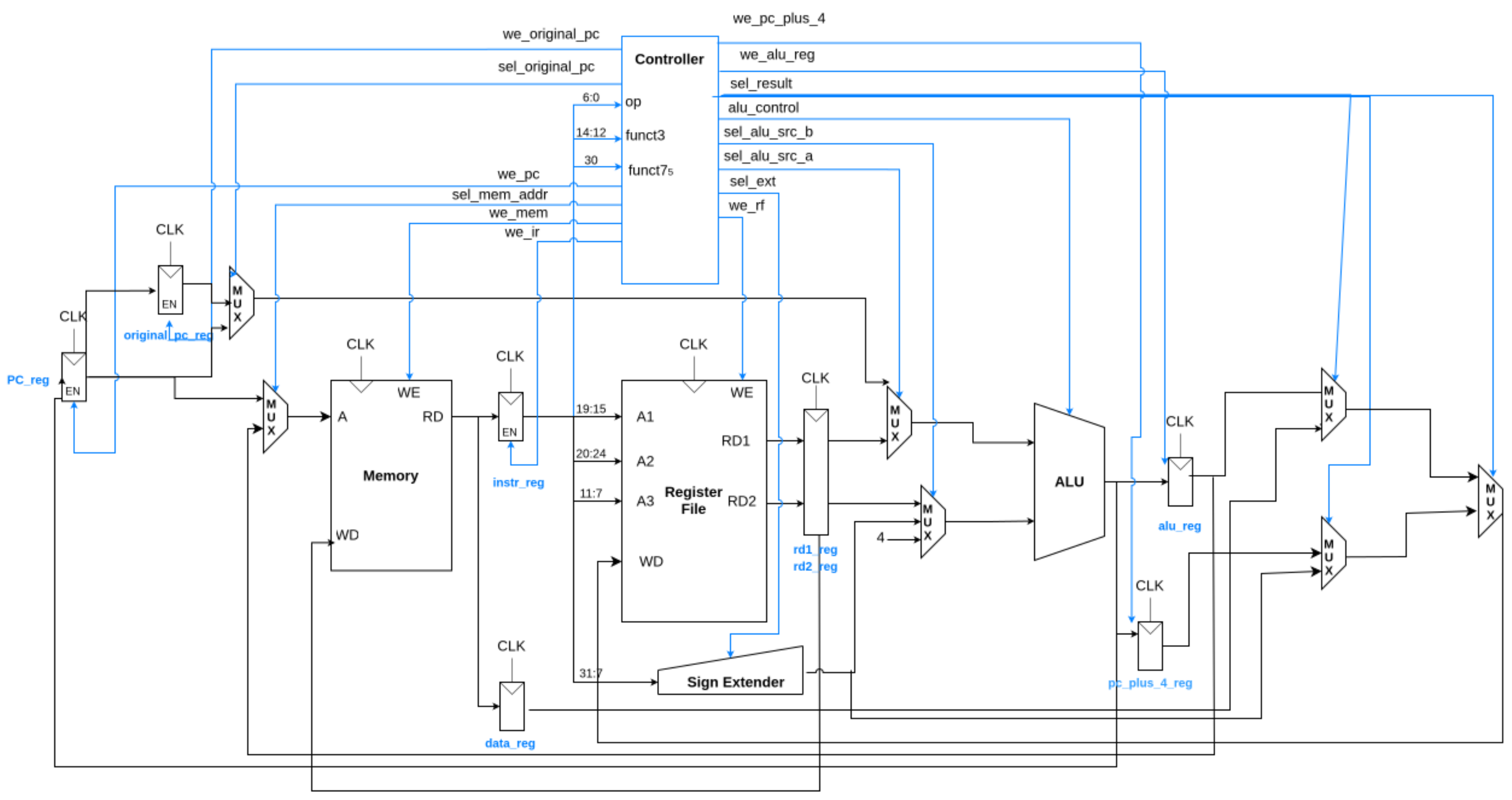
Michał Jakub Chrzanowski

Link to Github project: https://github.com/Mistryu/RISC-V-processor-Verilog/tree/RV_MC
Inside multi_cycle/rv_mc_drawio are the drawio files here are the diagrams:





Answers to questions in task 4:

In Figure 1, some registers have write enable (en) while some do not. Explain why en is necessary or not.
-> We need to prevent the registers from getting currupted. For example PC_reg captures pc only during fetch and jal and en prevents it from getting currupted in other stages which could japerdise the execution. Some registers like rd1_reg don't have it and are always enabled since rf reads update each cycle for data consistency.

Why is the output of the sign extender not registered?
-> Because it is needed in the same cycle and it is purely combinatorial with no internal state.

Where is the slowest datapath (critical path) between two registers, given the delays of the components are constants as shown in Table 1?

PC update path. It is the longest path and at the same time it's very critical to the operation.
PC_reg -> Memory -> instr_reg -> Controller -> ALU -> PC_reg

How does a multicycle architecture allow the clock period to be shorter than in a single-cycle one? What is the benefit?

-> in multicyle the clock has to be greater then the longest stage compared to the longest instruction path which is generally shorter. Thus shorter clock means faster execution. Also we can optimase each stage seperately and reuse components.

Why might a multicycle design be easier (or harder) to extend with new instructions compared to a single-cycle design?

-> It's easier because it's more modularised which makes it easier to extend. We don't have to handle the entire instruction at a time but just a part of it so since the instruction itself is seperated into stages it's easier to implement and test each stage seperately then to implement the entire instruction at once like we have to do in single instruction cycle.

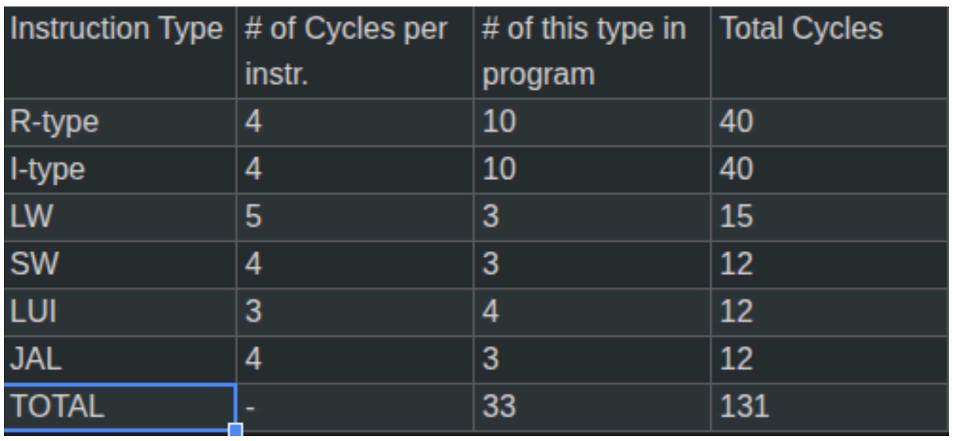
Briefly describe the bug you have encountered.
-> The processor was showing X (unknown value) in PC and all registers which prevented the execution. I have added a reset functonality in order to set them at the start which fixed the issue. There were also issues with lui where I had to add additional register and signal to make sure the command works and we have the value of PC + 4 saved.

Task 4.2
The numbers are calculated based on program.hex

Average CPI calculation:

CPI = 131 / 33 =~ 3.97 cycles per instruction

I got 131 total cycles in my code as well.



Instruction Type	# of Cycles per instr.	# of this type in program	Total Cycles
R-type	4	10	40
I-type	4	10	40
LW	5	3	15
SW	4	3	12
LUI	3	4	12
JAL	4	3	12
TOTAL	-	33	131