# Naziia Raitova – Work Journal

## Milestone 1

**December 19th, 2023**

- Collaborated with the team during scheduled class sessions, established communication channels, and determined the meeting schedule.

**December 21st, 2023 (~2 h)**

- Created the instruction set using Excel.
- We checked what kind of instructions we need and drafted the core instruction format with six different types. We also brainstormed on how to address issues such as including larger immediates and branches.
- Brainstormed to discuss and refine the register concepts.
- We made a list of registers and created a memory map, allocating space for text, data, and any special addresses required by our design.
- Added pages 1 - 3.

**December 28th, 2023 (~1 h)**

- Met with Luke via Teams. Luke and I wrote the Euclidean Algorithm in the google doc (+ comments).
- Luke and I went through the instruction set one more time to make sure we have everything we need.
- Added a few new instructions to make it work. Made some changes in the google doc.
- Added pages 5 - 9.

**January 9th, 2024 (~2.5 h)**

- We went through our Euclid's Algorithm assembly code to make sure we have everything right.
- I wrote the fragments in C and assembly code (+ comments).
- We then translated our assembly programs to the machine code, including 'relprime' and the fragments. Addresses and a machine code column were added to the Excel sheet.
- I focused on translating the 'gcd,' 'gcd_loop,' and a portion of the 'relPrime' functions.
- Tulsi, Luke, and I collaborated on translating the fragments' code into machine code.
- Added pages 9 - 11.

# Milestone 2

**January 13th, 2024 (~2 h)**

- We worked on implementing the components descriptions together.
- I wrote inputs and outputs for Immediate Generator, Control Unit, ALU, and ALU control.
- ALU Component: Defined the inputs and outputs. We all together specified the operations the ALU can perform, such as addition, subtraction, bit-shifting, logical AND, OR, and XOR. Mentioned that the ALU operates on the rising edge of the clock (CLK).
- ALU Control Component: Inputs include input_instruction[14, 6-3], input_ALUOp, CLK[0:0]. The output is output_ALUcontrol[5:0]. The control unit interprets the instruction and sets the control signals for the ALU based on the operation specified in the instruction. I wasn't sure about input_instruction. The datapath from HW12 had instruction[30, 14-12] for the input for ALU control. [14-12] is funct3, but what's [30]??? Our design uses funct4 instead, and that's where [6-3] is coming from. Might need some changes later.
- Control Unit: Inputs (input_value[2:0]) suggest that it takes the opcode as input. Outputs include signals like output_branch, output_memRead, output_memtoReg, output_ALUOp, output_memWrite, output_ALUSrc, output_regWrite. Each is probably 1 bit, and I might add it later, but I wasn't sure if we wanted to change it. For now it's to be discussed with the group. The Control Unit is responsible for generating various control signals used in the overall processor based on the input values.
- Immediate Generator: Takes input_instruction[15:0] as input and produces output_imm[15:0] as the immediate value. It takes all 16 bits.
- Memory Component: We talked about how we wanted the memory to function, and we agreed to merge instruction and data memory. We chose to put instruction and data memory together to keep things simple and use resources efficiently. Doing this makes the design smoother and avoids the complications of managing separate memories for instructions and data.
- Initially, we all worked together to create the RTL design for a common set of instructions. This helped the team get a basic understanding of the RTL design. Afterward, we divided the various instruction types among the four of us. I specifically handled the Register Immediate (RI) instructions.

**January 14th, 2024 (~4 h)**

- We continued working on component descriptions and RTL design for each instruction.
- Focused on completing remaining tasks, specifically addressing Memory and PC. I worked on memory outputs.
- We decided to get rid of ALU control, assigning all tasks to ALU. I adjusted ALU inputs and outputs accordingly, including additional inputs for ALU like ALUOp.

- After addressing the remaining tasks, I revised the RTL design for RI instructions, specifically changing 'IG([15:0])' to 'imm' to generate immediates and maintain consistency with other types' designs.
- Worked on control unit inputs and outputs, adding some extra notes to clarity where the bits are coming from.
- We discussed and resolved issues with jalr and branch RTL design, emphasizing the logic behind addressing.
- I made a column called Notes for unclear aspects in the RTL design. Yueqiao suggested making separate columns for input notes and output notes. Finally, we all decided to put comments instead to avoid making too many columns.
- Worked on commenting PC outputs, Control Unit inputs + outputs.
- We separated Memory into Instruction and Data Memory, which Luke explained would simplify our work.
- Helped identify inputs and outputs for Instruction and Data Memory.
- Wrote Data Memory output and Instruction Memory input. Luke added more things after.
- Finally, we worked on revising our mistakes from Milestone 1. I wrote an additional snippet for recursion in C and assembly code (Snippet 3). Might need group revisions later.

## Milestone 3

**January 20th, 2024 (~2 h)**

1. Worked on drawing a multi-cycle datapath basically the whole time. Created a MC Component descriptions spreadsheet. Brainstormed on what kind of components we want to see in our datapath. Tulsi drew the datapath, while we all shared our ideas.

**January 21st, 2024 (~2h)**

1. I cleaned up the single and multi-cycle descriptions for all instructions. Reviewed and checked for errors.
2. Worked on the behavior and RTL Symbols of all components of multi-cycle component descriptions. We brainstormed on which instructions are going to have a clock edge, what additional inputs/outputs we need to have. Checked behavior for components again, deleted the clock for ALU and Control Unit.
3. Fixed some errors in our datapath, like MDR inputs and outputs.
4. I created a new component Instruction Register in our multi-cycle spreadsheet. Wrote inputs/outputs, behavior, and RTL symbols for it, while Luke wrote testing for IR.

5. Finally, cleaned up the single-cycle RTL for the instruction set.
6. We all worked on MC RTL for every instruction type. I was typing while we all shared our ideas.

**January 23rd, 2024 (~1h) - Class time**

1. I created the summary table for multi-cycle for all instruction types and individual instructions (like branches, jal, lui, jalr). Yueqiao and I found some errors in our RTL and worked on them. The hardest part was implementing special instructions. We decided to add ALUOut = ALU(+, PC, 2) to our first cycle instead (it was on our second cycle, like in the summary table Dr. Williamson gave us). Yueqiao explained that we can do that on the falling edge, and I trusted his judgment. Found some errors in jalr RTL and completely changed it.

## Milestone 4

**January 28th, 2024 (~2h)**

I helped Luke and Yueqiao draw the state machine on the whiteboard, primarily helping Luke with 2RI, RI, 3R, and lw/sw types. I also fixed the RTL and added notes for clarification. We brainstormed on how to approach branchType: are we using the ALU for that? Tulsi created an electronic version of the state machine.

## Milestone 5

**January 30th, 2024 (~2h)**

Divided the responsibilities. Luke and Tulsi are doing lab 7. Yueqiao is doing a control unit verilog file. Luke is doing tests for Immediate Generator, Tulsi is doing tests for PR file and SimpleRegister file. I'm doing the tests for ALU, PC, IR.

**February 2nd, 2024 (~2h)**

Compiled the tests and tested all components, figured out ModelSim. Found out that some of our components have errors.

**February 4th, 2024 (~2h)**

Divided the datapath into 3 parts: FetchAndMem (Luke's part), Data (Tulsi's part), and Calculations (my part). Yueqiao worked on the control unit.

**February 5th, 2024 (~2h)**

Yueqiao worked on the control unit, while Luke, Tulsi, and I wrote the verilog files for our "bubbles", and compiled them to make sure we don't have syntax errors. We had some issues and I was confused why quartus was giving errors if we used output reg, but if we write output wire, it worked just fine, so we left it like that. For my part (Calculations.v), I wrote mux2 and mux3 myself, but it needs to be changed to generating it from the existing files Yueqiao created. I also revised and compiled Tulsi and Luke's verilog files.

We agreed to work separately on remaining tasks: Yueqiao – control unit; Tulsi, Luke and Naziia - tests for each group.

**February 8th, 2024 (~2h)**

Luke and I finished the tests for our bubbles (FetchAndMem and Calculations), but Luke couldn't write the test cases for his bubble (FetchAndMem) since the lab7 is still not done. Yueqiao and Luke worked on lab7 during the meeting, while I was writing the tests for the third bubble (Data_tb). It was a little challenging since I had to write tests for each type and before writing the actual test cases, I had to put values into some of the registers, so we could actually get the values from the registers while we did the testing. I didn't know how to do it, and Luke helped me with that part.

February 11th, 2024 (~4-5h)

Luke and Yueqiao demod the lab, so we could finally piece some things together. Started compiling and testing all the components to make sure each one of them works. I worked on ALU, PC, and Simple Register. I kept running into errors in PC and couldn't figure them out for 2 hours, so Yueqiao helped me to finish it. We were also assigned to test and compile our “bubbles”. I tested Calculations, but it had lots of compilation errors in it, so it took some debugging. Luke and I compiled and tested Immediate Generator. We had a problem with merging lui and immediate, but in the end, it worked.

February 13th, 2024 (~1.5h)

I started working on building the Verilog file for the whole processor. Data and FetchAndMem are not done yet, so it was the first ‘draft’ of The Lime.

February 14th, 2024 (~2h)

Yueqiao changed the datapath (added the additional 2-3 muxes) because he found an error with jalr. I had to change the whole Calculations part since we added additional inputs and outputs. Luke and I finished Immediate Generator, and it worked 100%.

February 15th, 2024 (~2h)

I started fixing the ‘Data’ bubble since we had a meeting with Dr. Williamson on Friday, and I wanted to have at least a good structure for the processor. Added extra muxes it was lacking and made it compile in about an hour. We changed the test cases according to new changes, but it wasn’t exactly passing any tests.

February 16th, 2024 (~4h)

Luke and I met an hour before the meeting to finish up the ‘Data’ section, but I discovered that now Calculations was failing. Yueqiao changed the memory map. We had a meeting that evening, so I tried to fix Calculations. Fixed Calculations section by the time Luke finished the Data section. We met later that day, and I started officially fixing the Verilog file for the processor since we had all components ready. Luke finished I/O and with Dr. Williamson’s help we wrote a test bench for the processor, while Yueqiao was finishing up the Control Unit.

February 18th, 2024 (~5h)

Spent the whole time fixing the processor file. Yueqiao was done with Control Unit, so it was really all we needed to do. Tulsi was working on the Assembler, but I think Yueqiao was doing the Assembler as well.

February 19th, 2024 (~5-6h)

Met with Dr. Williamson, and he helped us with debugging. It was a long process of finding small errors and trying to fix them. We would spend an hour trying to figure out a problem and an hour trying to solve it. At some point, I think we had a problem with sw, but actually it was a problem with branches because addressing was working incorrectly and we would just miss some of the instructions. When Luke and I managed to fix all the errors, it actually gave us an output and not just xxxx. However, it was a wrong output, but at least we knew our processor was working.

February 20th, 2024 (~6h)

Luke and I met with Dr. W and were working on debugging the processor. While Yueqiao was finishing up the assembler, Luke and I went through the whole algorithm code and found out that the whole thing is completely wrong. The reason why it was giving the wrong output is that we would return the final output when we would finish gcd, not relPrime. The processor worked, but we had to completely change the relprime code to what we thought was right, but after that, the processor wouldn't stop and would just give all xxxx all over again. I started writing relPrime algorithm code basically from scratch, while Luke and Yueqiao were making sure we didn't have any errors with branches. Luke and I had to go through each line of code and check the addresses and the values of every single register to make sure it's going in the right direction. At some point we found an error where we had bge, instead of bgt, so we implemented bgt using bge and beq. At the end, it worked, but it would give all 0x0000 because we weren't storing m value in the stack. When we fixed that, it worked and we were all done.