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.