Design Journal

MILESTONE 1 WORK:

Sunday, April 3, 2022

Working on the Milestone 1 [1h] + team meeting [2h]

- We decided to build an accumulator based processor. We designed all the instructions, the instruction types, and the calling procedures.

Wednesday, April 6, 2022

Finalize the document [15 min]

- Specify the calling procedures.

MILESTONE 2 WORK:

Sunday, April 8, 2022

Working on the Milestone 2 [30min]

- We split the work, get updated with each other, and figured out the basic design idea of Milestone 2.

Wednesday, April 19, 2022

Revise the RTL [15 min]

- Double checked the work Helen had done and get a better understanding of our proposed RTL design.

Wednesday, April 20, 2022

In Class Project Meeting [1h15 min] + Evening work [4h]

- Specified the components we are going to use,and wrote the assembler program. Also finalized the document.

MILESTONE 3 WORK:

Friday, April 19, 2022

First thoughts on the datapath [50 min]

- We designed the very first few parts of our datapath, and decided what next time we should meet. At this stage, we simply started from the multicycle datapath, and added a few components as we went through the RTLs & instructions. We adopted this Top-down way of designing so that we are not missing any part.

Sunday, April 24, 2022

Design the RTL datapath [2h]

- We had the team meeting from 5:40 to 6:40 to work on the milestone 3. We decided to make the datapath multicycle. We tested all the instructions & their RTLs. Also, we figured that we will put our register on write first and then read, so that we don't need to add another register to store the value of our ALU.

Tuesday, April 26, 2022

Revise components design [30 min]

- I updated the components design from the last milestone, to specify the name of each input and output wire, so that it is easier for other team members to reference it in the later parts.

Wednesday, April 27, 2022

Evening work [3h]

- Implemented the 6 components in the Quartus with Verilog. The 6 basic components include the 16-bit ALU (alu16.v), 16-bit Memory (mem16.v), 16-bit mux with 1-bit opcode (mux1b16.v), 16-bit register (reg16.v), 8-bit to 16-bit sign-extended (se16.v), and 8-bit to 16-bit zero-extended (ze16.v). Aslo, I complied and figured all these components to be logically correct. And I renewed the Python-based assembler to match with our last update of the memory decision and new jump instruction and J type.