Ethan Wilson, WORK LOG

MILESTONE 1 & 2 WORK:

Wednesday, April 3, 2024

Met with team [3 hours]

We decided on making a memory-to-memory processor. It will take a 104-bit architecture however the programming will be much easier than other architectures because there are only 4 types of instructions. It will also be easy to implement because there is a lot of overlap between instruction types. We also allocated space at the start of the stack to store special values like zero, pseudo-instructions, function arguments and returns, the stack point and the program counter. We created all our instructions and how they would translate to machine code as well.

Thursday, April 4, 2024

Met with team [2.5 hours]

Decided to switch designs, still memory to memory but referring to memory will be offsets from sp instead of addresses. There are 4 registers for special values. Also worked on assembler in Java independently.

Saturday April 13, 2024

Met with team [3 hours]

Worked through instruction types and memory allocation for new design. Decided to have one sp register that the user cannot access and have all other special values be stored in each functions individual stack space.

Sunday April 14, 2024

Met with team [1 hour]

Worked through more instruction types and how different goals will be achieved such as calling conventions.

Monday April 15, 2024

Met with team [2 hours]

Started writing the design document and designing hardware. Created calling conventions for function arguments, where the caller will put the argument directly into the callees stack space before changing sp.

Tuesday April 16, 2024

Met with team [1.5 hours]

Decided to have a temporary sp register that gets updated instead of sp when the callee adds to sp so that the branch comparison still has the correct sp and then the branch automatically updates sp with temp sp after being called. Created a subsp instruction that will directing change sp so that a “fantom” branch is not needed to update sp after returning from a function.

Worked on assembler [3.5 hours]

Finished writing the assembler in Java so that we can write our instructions in a text file and it will automatically be assembled into machine code.

Wednesday April 17, 2024

Met with team [2 hours]

Created the RTL for the design and resketched the hardware. Decided that multicycle would be necessary since the branch instruction takes significantly longer than any other instruction due to the extra automatic functions that we implemented to make function calls programmer friendly.

Wednesday April 24, 2024

Met with team [2 hours]

Reworked the Datapath with control. Remade the components list and updated the RTL.

Thursday April 25 [4 hours]

Created the state diagram for the control and reworked it so the muxes could be combined with other signals to keep the logic down. The only problem I ran into was PCWRITE and MEMWRITE because they need to be anded with the cmp, but on some cycles they shouldn't be, so I added the SKIPCMP signal so that on cycle 1 PC can still always write and on cycle 2 during make mem can always write. So the logic will work like (MEMWRITE && (cmp | | SKIPCMP)) for memory and (PC && (cmp | | SKIPSMP)) for writing to the PC.

Saturday April 27 [2.5 hours]

Worked on implementing the control in Verilog.

Monday Aprill 29 [4 hours]

Completed control and worked on the ALU in Verilog.

Friday May 03

Met with team [2 hours]

Checked Verilog tests that had been completed and worked on new ones.

Wednesday May 8 [4 hours]

Completed ALU and started Memory in Verilog.

Monday May 13

Met with team [2 hours]

Worked on documentation as well as Verilog tests.

Tuesday May 14 [8 hours]

Completed memory and started working on full processor.

Wednesday May 15 [9 hours]

Completed full processor. Reworked Assembler in java and created the final testbench. Successfully ran Relprime in ModelSim.