CSSE 232 COMP ARCH 1 JOURNAL

TEAM ORANGE:

Drew Kilner

M3(datapath design):

Registers

10/12/23: (1hour) Began datapath drawing in online software. Decided to output 3 arguments from the register file due to needing rs1 and rs2 regularly, but also rd is needed for the input data in our store word instruction.

Pipeline implementation

10/13/23: (2 hours) Continued datapath implementation. Started the pipleline type implementation of the datapath. Split memory into instruction and data. Started the pipeline register groups. Put our PC incrementor in the decode stage so that the PC relative branching is not effected by it.

Control

10/16/23: (1 hour) Datapath implementation of control and added registers in the pipeline blocks for each signal needed per cycle. Currently computing 7 control signals, write reg, alusrc, aluop, branch, memwrite, memread, and a regstore(decides what to write to the register).

Forwarding unit

10/17/23: (1 hour) Datapath implementation of the forwarding unit. Added new registers in the pipeline block to send the rs1, rs2, and rd registers for forwarding conflicts. This will fix most problems with registers being needed right after one is operated on. Still have issues when loading a word and then using that register. Must put a bubble and hold the registers when doing this. To prevent have better coding educate.

Branching changes and branch prediction

10/18/23: (2 hours) Changed the datapath for branching. Moved the branching to happen inside the decode stage and not the memory stage. Added muxes for ALU sources 1 and 2 from the forwarding unit, and left the second mux for the imm vs register ALUsrc control. Forwarding unit sends a 2 for normal operation, 1 for write back data, and 0 for memory data. Decided that we are not going to implement a BDS, but we are going to create some simple branch prediction logic stemming from if branched before, branch now, then correcting this with a flush if incorrect.

M2/M1:

9/27/23: (1 hour) Brainstorm what type of architecture we want. Then we landed on an 8 register design. Possible future problems could be not enough registers. Created all the instructions needed to run the algorithm. Key takeaways: focusing on speed, running Euclid algorithm as fast as possible, ended on Load/Store.

10/4/23: (after meeting): (30 minutes) Made ways to change our project from being a 16 bit RISC-V. Key takeaways: changed all symbols for instructions, rearranged it to rd operation rs1, rs2 (a = b + c type of structure).

10/4/23: (15 minutes) Updated symbols in existing document and found some new ideas and changes nothing major.

10/6/23: (2 hours) Created basic examples of functions. Translated from java type code to assembly to machine code. Key takeaways: still felt like RISC-V maybe find some things to change about it later.

10/8/23: (45 minutes) Changed symbols and instructions in my parts of the document to match new instructions. Key takeaways: now feels like our own project no longer RISC-V in 16 bits.

10/8/23 (3 hours) - Created the RTL path for M-Type (loading and storing memory) instructions and the 2 first base instructions for all types. Then started the table of components. Key takeaways: 2nd step creates 3 arguments from rs1 rs2 and rd for later use due to needing those based on our use of instruction types.

10/11/23 (1 and a half hours) – Edited RTL for implementation of pipeline. Added registers we need for pipelining and split memory. Planned for Milestone 3. I will be taking control of the datapath and planning out the pipelining. Key takeaways: RTL is now mostly set for pipelining, and I will be the main outlook for the pipelining.