Workplan: Floating Point Instruction Decode

ToDo List:

1. Research the device/spec
    a. Find learning resources (eg lecture slides, youtube videos, standards, etc) and list them in the table
2. Fill out this sheet (functionality, needed module ports, detailed block diagram, how to interact w/ module, etc.)
3. Use this sheet to create a spec and test plan
4. Once your spec and test plan are done, start writing RTL
    a. Update your spec and test plan as things change
        i. Be proactive please
    b. Don't start writing RTL until 1-3 are done
5. Finish RTL, write sanity check test bench
    a. Basic checks for the test cases you identified in step 3
6. Run the synthesis prototyping flow
    a. Confirm no latches, no obviously wrong netlists (eg not having the expected amount of registers, submodules not included in the final netlist, etc)
7. Run the synthesized netlist through your sanity check testbench
    a. Also through DV test bench if it exists by this point

Approximate Timeline:

1. Have steps 1 and 2 done/in progress by end of 2025 semester
2. Have step 3 done and step 4 started by end of September, Fall 2025
3. Have step 5 done by January-ish, Spring 2026
4. Have step 6 and 7 done by Feburary-ish, Spring 2026

Resources:

Source	Description
https://www.youtube.com/watch?v=dQhj5RGtag0	Video discussing floating point
unpriv-isa-asciidoc.pdf	RISC-V Spec, use F extension
https://ece2020.ece.gatech.edu/readings/datapaths/index.html	More information on each block in the diagram

Functions:

- The overall functionality of the Floating Point Instruction Decoder is to derive which of the Floating Point functions our input is referring to. This will happen in the Decode phase. We will be using the F extension, which will be introducing 32 floating-point registers, f0-f31, each 32 bits wide, and a floating-point control and status register fcsr. The floating-point instructions include the following: Single-Precision Load and Store instructions, Single-Precision floating-point computational instructions, Single-Precision floating-point conversion and move instructions, Single-Precision floating-point compare instructions, and Single-Precision floating-point classify instructions.

IO Ports (add more to this as needed!!):

- List out the ports your module needs to operate
- What are your inputs? What does your module output?
- Input :
- instr (32bits, the instruction we are decoding)
- reset
- clk
- Output :
- fp_op (output referring to the operation we derived from input)
- rs1, rs2, rs3 (source register indexes, three needed in case of FMADD.S, FMSUBB.S, FNMADD.S, FNMSUB.S)
- rd (destination register index)
- eff_addr (effective address, used in load and store operations, refers to the address in memory being stored in/loaded from)
- eff_write (effective write enable, high in store operations when effective address is being writen to)
- eff_read (effective read enable, high in load operations when effective address is being read from)
- fp_write (high when one of the registers is being written to)

- fp_read (high when one of the registers is being read from)
- rm (floating point rounding mode)

Block Diagram: