Overview:

RISC-V is an open-source instruction set architecture developed by UC-Berkeley and has been picking up popularity and community over the years. Apart from the open-source instruction encoding, there are also developing tools such as C compiler, simulator, etc., available. I have an ongoing project called Riscy which is to write my own risc-v core with the ultimate goal to run a primitive Unix operating system (such as xv6) on a FPGA loaded with my risc-v core. I'm planning to use one of my future milestones of this project as my semester project and develop an arm-based debugger for the core to fulfill the software requirement for this project. The project is called Riscy because I morphed RISC-V’s V to Y, so it makes easier for me to read. (And because I can see it will consume me hundreds of hours building this project and it’s a ris-ky decision) Github link:

Current status of project Riscy:

The core is now able to run basic C-compiled machine code on functional simulation, with a 1-cycle delay instruction / data memory. There’s still some very obscure edge case that might lead to bug as the random instruction test failed.

My Planned steps for project Riscy in the semester:

1. Use proper AXI-Lite interface, indefinite-delay ram for the core instead of a fixed 1 cycle delay (on simulation)
2. Use AXI-Lite bus and interconnect to connect everything (cache, memory, bootloader, etc.)
3. Map everything to the de-1 development board, and use the tiny on-chip memory for now
4. Fulfill the timing requirement for the core, and run a very basic program (such as blink a led using memory-mapped IO)
5. Develop a program that enables the Arm core on the de-1 board to have access to the AXI-Lite bus on the risc-v core, and can perform basic write / read operations to the risc-v memory space via AXI-Lite bus
6. Polish the program so that I can use it as a debugger / data injector to my risc-v core, and able to interact with it using uart serial connecting the de-1 board to the user console.
7. (Hopefully) develop an interface that enables the risc-v code to access the hardware memory (sdram or DDR3) so that I can run larger programs

Timeline:

Each step mentioned above should hopefully take around a week to finish