**RISC-V FYP**

* ISA Spec: <https://riscv.org/specifications/>
  + Includes both the User-Level and Privileged Spec.
* Based on Fearghal’s suggestions, ideally the core should implement the base integer instruction set, RV32I with machine mode privileges from the privileged ISA specifications. The core should be pipelined to augment the existing course content.
* SiFive, founded by the inventors of RISC-V now generate RISC-V cores and other open source RISC-V related IP which may aid in the FYP:
  + Bus Fabric Spec:  
    <https://www.sifive.com/documentation/tilelink/tilelink-spec/>
    - Very powerful communications protocol for memory accesses. Designed with RISC-V in mind.
  + Debug Spec: <https://www.sifive.com/documentation/risc-v/risc-v-external-debug-support/>
    - Another spec designed for RISC-V cores, started out as a mailing list and was then adopted by SiFive. Very powerful spec, allowing one debugger module to handle thousands of RISC-V harts (hardware threads).
* Books: <https://riscv.org/risc-v-books/>
  + Currently own “The RISC-V Reader” and “Computer Architecture: A Quantitative Approach”. Both very useful and very informative texts.
* Possible additions depending on success of implementing the base ISA:
  + TileLink Memory Interface
  + Interrupt Controller
  + Debugger
  + Adding L1 Data and Inst. Caches with a L2 Cache
  + Adding “M”, “C” and possibly “A” extensions
* Floating point operations from the “F” and “D” extensions would alone be sufficient in complexity for a post-graduate project, the work required to extend the integer pipeline to compensate for floating point operations is not feasible for an undergraduate final year project.
* Useful Links:
  + Graphical RISC-V Pipeline for teaching: <https://github.com/mortbopet/Ripes>