NAME?

MILESTONE 1 WORK:
Monday, January 8, 2024:
- Refactored/finalized instruction set [1.5 hours]
- Determined instruction types [1.5 hours]
- Completed Instruction Set table [1 hour]
Tuesday, January 9, 2024:
- Wrote assembly examples for each instruction [1 hour]
- Translated relprime into machine code [1 hour]
- Wrote example common operation snippets [30 min]
- Finished up memory map, made modifications to expanded instruction set table [30 min]

MILESTONE 2 WORK:
Thursday, January 11, 2024:
- Ethan gave hardware demo in Sandbox: showed locations+functionality of main register bus, special dedicated registers (sp/ra/pc)
  - Since instructions all have immediates in same location but of differening length, immediate genies for imm. instructions handled right after decoder, MUX selects them
- Began drafting document with RTL instructions, completed 10W
Friday, January 12, 2024:
- Ethan and I worked on refactoring document.
  - We decided on recursive calling conventions: local vars that need to be preserved between calls should be backed up+restored in saved registers, since this is more of a "callee responsibility" Cool
  - Assembled example programs [20 mins]
  - I redid the memory map, Ethan fixed the example program instructions to follow it
  - Decided to refactor instruction formats to table format for readability
  - Examined verilog parts we plan to use: using equality that xors to a standard bitwidth reg rather than alu subtraction

Monday, January 15, 2024:
- Completed RTL for RETURN [10 min]
- Drafted component list [2 hrs]:
  - We decided (for the time being, we will likely refactor into multicycle in the future and change things)
  - there will be 4 separate ALUs:
    - One for simple 2-input logic instructions (Add, and, etc.)
    - One for branch instructions, as they output a single-bit flag + use gates rather than adders to do comparisons
    - One for PC/SP modification instructions, so output can be wired directly to pc/sp/ra
    - One for other instructions, that will likely be divided up in the future (since each of these instructions uses their own logic)
  - Massive instruction decoder determines input flags for ALUs (might refactor this into opcodes in the future/clean it up)

why just not build one?

- Cleaned up RTL based on new component list: fixed naming conventions to more clearly demonstrate what components were being used (we were using A/B/C letter variables for everything when these are reserved for loaded regs) [30 min]
- Drafted RTL error checking process and fixed some RTL errors based on it [30 min]