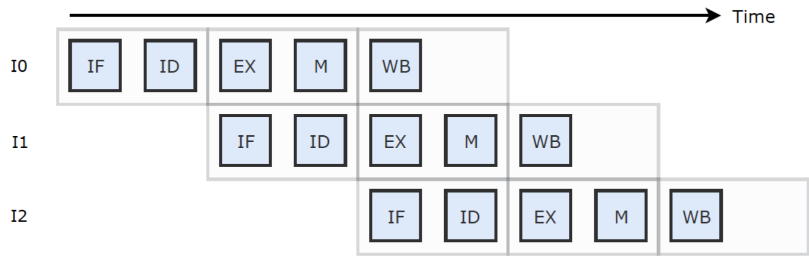
CSCE 3301 – Computer Architecture
Fall 2025

Project 1: femtoRV32
RISC-V FPGA Implementation and Testing

1. Requirements

You are required to implement a RISC-V processor and test it on the Nexys A7 trainer kit. Your implementation must satisfy the following:

1. It must support the RV32I base integer instruction set according to the specifications found here: https://riscv.org/specifications/ratified/. All **forty-two** user-level instructions (listed in Chapter 35 pages 609-610 of the "The RISC-V Instruction Set Manual Volume I: Unprivileged Architecture" (Version 20250508) and explained in Chapter2 of the same document) must be implemented as described in the specifications except `ECALL`, `EBREAK`, `PAUSE`, `FENCE`, and `FENCE.TSO` instructions. Instead, your implementation should interpret all of these 5 instructions as halting instructions that end the execution of any program (by preventing the program counter from being updated anymore).
2. It has to be pipelined with correct hazards handling.
3. Your implementation should use a single memory for both data and instructions. The memory should be single ported and byte addressable. The single memory constraint introduces structural hazards that degrade the CPU performance and increase the effective CPI drastically. One solution, is to issue an instruction every 2 clock cycles (effective CPI=2). By doing so, memory accesses of different instructions will never occur on the same clock cycle.



Pipelined CPU with Every-other-Cycle Instruction Issuing

   This can be seen as the CPU executes each instruction in 6 cycles divided into 3 stages. Each stage uses 2 clock cycles (C0 and C1).
   - Stage 0: Instruction Fetch (C0) and Registers read (C1).
   - Stage 1: ALU operation (C0) and Memory read/write (C1).
   - Stage 2: Register write back (C0). C1 is not used by this stage.
4. You should provide a set of test cases demonstrating the full support of all instructions and all hazard scenarios. Optionally, you might use code segments from the RISC-V official compliance test suite. It can be found at: https://gitlab.com/incoresemi/riscof/-/tree/master/riscof/suite/rv32i_m/I

Important Note: You will be provided with a set of Verilog descriptions that you can use to model the ALU, the Immediate generator, and a general constant definitions file. Make sure to understand each of the files perfectly to be able to use them appropriately.

2. Bonus Features

To get bonus marks, you can implement up to 2 bonus features from the list below:

1. Add support for compressed instructions to effectively support the full RV32IC instruction set except for compressed instructions that do not map to supported instructions according to the requirements above. The compressed instructions are also described in the official specifications mentioned above.
2. Add support for integer multiplication and division to effectively support the full RV32IM instruction set. The integer multiplication and division specifications can also be found in the document above.
3. Implement a 2-bit dynamic branch prediction (and branch target address prediction) mechanism
4. Move the branch outcome and target address computation to the ID stage and handle the resulting data hazards
5. Provide another solution to handle the structural hazard introduced by using a single single-ported memory (other than the solution suggested above).
6. Provide a test program generator. The generator (written in any language of your choice) should be capable of generating a sequence of random but valid instructions to use as a test program for your implementation.

3. General Guidelines

- Work with the same lab partner you work with in the lab. A team leader must be selected. She/he will be responsible for submitting all the deliverables on blackboard. Any group member may interact with the course instructor and TA for any technical issue.
- Every group member must log all of her/his activities in a journal (a text file). A journal entry may look like the following:

```
November 4, 1:55PM: finished the forwarding unit.
                Fixed issue in pipelining registers.
```

- Every deliverable (except for the first milestone) must be submitted by the group leader as a single zip file. The zip file must include the following:
    - A readme.txt file that contains student names as well as release notes (issues, assumptions, what works, what does not work, etc.)
    - A journal folder that contain the journal file of each team member.
    - A Verilog folder that contain all Verilog descriptions
    - A test folder that contain any files used for testing
    - A PDF report documenting your design (including a schematic of the proposed datapath), your implementation, any issues encountered, solutions, screenshots of test waveforms, etc.
- You must follow the recommended coding guidelines (posted as a separate document).

4. Deliverables

- MS1 (Done): Team member names, student IDs.
- MS2 (Thursday November 6): A single cycle datapath block diagram and Verilog description supporting all of the RV32I instructions as described above. Also, basic test cases covering all supported instructions should be included. At this stage, 2 separate memories must be used for instructions and memory. No thorough testing is required at this stage. Testing on FPGA is optional at this stage.
- MS3 (Thursday November 20): Pipelined datapath block diagram and Verilog description. Your pipeline design must use a single single-ported memory for both instructions and data. Final implementation and report including proof of thorough FPGA testing (covering all 42 instructions and hazard situations) and any bonus features you intend to submit.
- Demo (TBA): Each team will reserve one of the available time slots in a shared Google sheet to present a quick demo of their work. During the demo, each team should be prepared to explain and run at least one of their provided test cases. The team should also be ready to answer any questions about their implementation and report.

5. Grading Criteria

- This project is first of 2 projects. Both projects collectively are worth 40% of the course marks. This project will count for 25% of the course marks while project 2 will count for 15%.
- The project will be graded out of 100 points distributed as follows:
    - MS1 deliverables: 5%
    - MS2 deliverables: 15%
    - MS3 deliverables: 80%
    - Bonuses: Each bonus feature will count for 5% with a maximum of 2 bonuses worth 10%.
    - Deductions:
        - -5% for not complying with the posted coding guidelines.
        - -5% for not following the required submission directory structure.
        - -5% for every late milestone submission (one day maximum)
        - -100% for plagiarized submissions.
    - Group members may receive different grades based on their contribution to the project