

THE AMERICAN
UNIVERSITY IN CAIRO
الجامعة الأمريكية بالقاهرة

CSCE 3301: COMPUTER ARCHITECTURE

Lecture 13: Project Description & Support
Dr. Cherif Salama

PROJECT: FEMTORV32

- Same teams as lab teams
 - Tasks must be equally divided
- You will design, model and test a pipelined RV32I with a single memory for data and instructions



GRADING AND GENERAL INFO

- This project is the first of 2 projects for the semester. It will count for **25% of the course marks** divided as follows.
- The project will be graded out of 100 points distributed as follows:
 - **MS1 (team formation): 5%, done**
 - **MS2 (single cycle, 40 instructions, separate memories) : 15%, Mar 28**
 - **MS3 (pipelined, single memory): 80%, Apr 8**
 - **Demo, time slots to be announced**
- Bonuses: Each bonus feature will count for 5% with a **maximum of 2 bonuses worth 10%**.
- **Deductions:**
 - -5% for not complying with the coding guidelines.
 - -5% for not following the required submission directory structure.
 - -5% for every late milestone submission (1 day maximum).
 - -100% for plagiarized submissions.



DEVELOPING TEST CASES

- You can use the RISC-V official compliance test suite
 - <https://github.com/riscv-software-src/riscv-tests/tree/master/isa/rv32ui>
- You can develop your own test cases, I recommend using the RARS simulator
 - Save the generated machine code in hexadecimal in a text file with a “.hex” extension
 - Load the hex file into your memory (Verilog array) using the Verilog system task **\$readmemh ()**

```
reg[7:0] mem[(4*1024-1):0]; // 4KB memory

initial begin
    $readmemh("./hex/test2.hex", mem);
end
```



SUBMISSION FILE STRUCTURE

- **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 Report folder** that contains your reports.
 - One report PDF should be added with each new milestone. Therefore, this folder should contain two reports in the final submission.



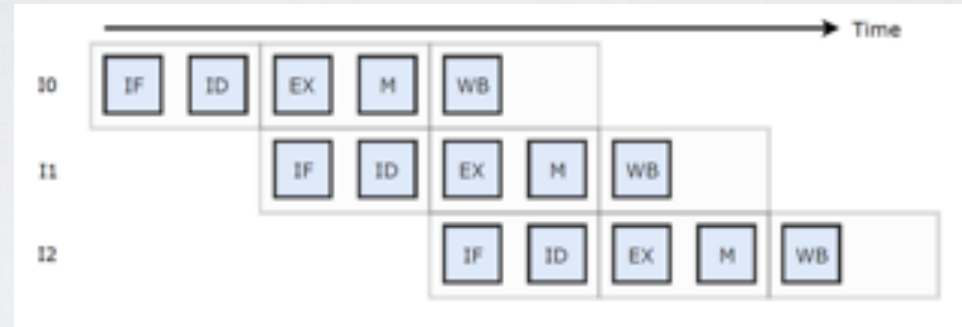
IMPLEMENTING CONDITIONAL BRANCHES: BEQ, BNE, BLT, BGE, BLTU, BGEU

- **Subtract** and check the following flags:
 - Zero: Z, Carry: C, Overflow: V, and Sign: S
 - BEQ: Branch if Z
 - BNE: Branch if $\sim Z$;
 - BLT: Branch if $(S \neq V)$
 - BGE: Branch if $(S == V)$
 - BLTU: Branch if $\sim C$
 - BGEU: Branch if C
- **Note:** All the flags above are generated by the provided ALU Verilog module.



PIPELINE WITH SINGLE MEMORY

- Structural Hazards introduced by single memory constraints can be removed by **issuing one instruction every 2 cycles** as follows



- Also, **this simplifies the forwarding logic** as the only possible RAW data hazard can occur because of data dependency between two adjacent instructions and allows the handling of load-use hazards as normal RAW hazards that can be resolved using forwarding
- You may think of this new pipeline as a 3-stage pipeline where every stage is constructed out of 2 old stages: [IF ID], [EX M] and [WB --]



BONUS FEATURES

- Add **support** for **all compressed instructions** to effectively support the full RV32IC instruction set.
- Add **support** for **all integer multiplication and division instructions** to effectively support the full RV32IM instruction set.
- Implement a **2-bit dynamic branch prediction** mechanism
- **Move the branch outcome and target address computation to the ID stage**
- Suggest and implement a **different solution** to handle the **structural hazard** introduced by the single memory requirement
- Provide a **test program generator**

