HW #4: Pipelined CPU

國立陽明交通大學
NATIONAL YANG MING CHIAO TUNG UNIVERSITY

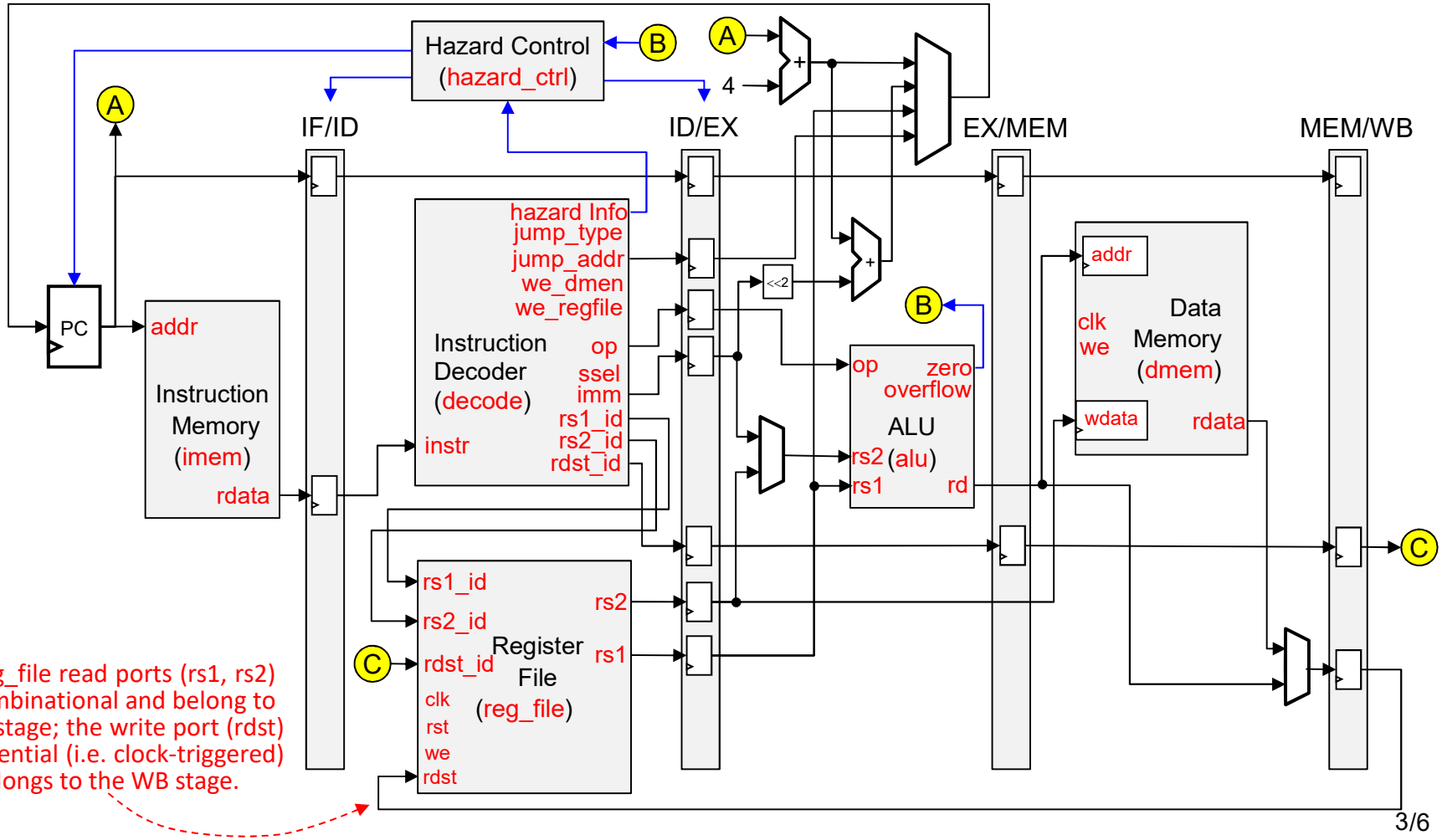
Chun-Jen Tsai
NYCU
04/29/2024

HW #4: Pipelined CPU

- Goal: modify the single-cycle CPU from HW#3 into a pipelined CPU
  - You shall neither implement forwarding nor branch prediction
  - The key point is to handle hazard control properly in this HW
  - For data hazard, you must properly stall the pipeline stages
  - For control hazard, the branch condition must be evaluated at the execute stage, and flush the pipeline when necessary
- The deadline of the HW is on 5/23, by 5:00pm.

Simplified Pipelining Diagram

- All wires between pipeline stages should go thru regs:



Hazard Control Unit

- The hazard control unit should stall the pipeline by inserting bubbles to stall the pipeline
- Please read the textbook carefully to understand how to implement pipeline control
  - Note that we neither implement forwarding nor branch prediction in this HW
  - For data hazards, just stall the pipeline
  - For branch hazard (control hazard), evaluate the branch condition at the execute stage, and simply flush the pipeline upon hazard and start fetching instructions from the new target address

Guidelines for HW#4

- Your top-level block diagram shall be instantiated using a Verilog module `core_top.v`
  - The templates of `core_top.v` from HW#3 should be used.
  - Do not modify the module interface of `core_top`. TAs will use a different testbench to test your `core_top.v`.
  - The block-diagram synthesized by your `core_top.v` does not have to be exactly the same as the one shown in page 3.
- The I/O ports of the `hazard_ctrl` module are not defined, you can freely define your own ports.

HW #4 Grading Guide

- You should upload all your code to E3 by the deadline
- To evaluate your design, the TAs will use different programs to test your CPU.
  - The program will do some computations and write the result to the data memory
  - At the end of execution, the data memory and the registers should contain the correct result