

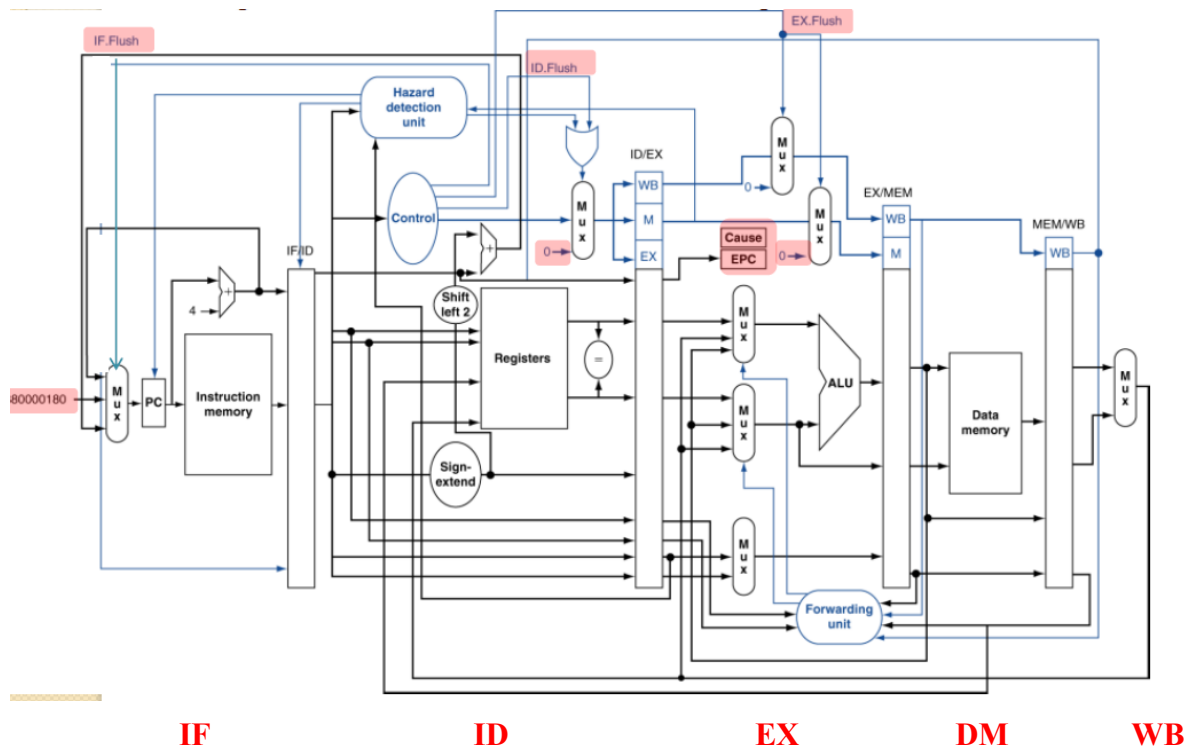
Project #2

1. Project Objective

- Implement a pipelined, functional processor simulator for the reduced MIPS R3000 ISA, following the specification “*Datasheet for the Reduced MIPS R3000 ISA*” in *Appendix A*
- Design your own test case to validate your implementation, particularly on hazards handling.

2. Project Description

a. Pipeline Description



- 5 stages in pipeline: instruction fetch (IF), instruction decode (ID), ALU execution (EX), data memory access (DM) and write back (WB).
- Conditional branches and unconditional branches are determined during the ID stage.**
- Load/store computes the address to be accessed in D memory during the EX stage, and accesses data memory during the DM stage.**
- Arithmetic/bitwise shift/logical operations are done during the EX stage.
- There are two forwarding paths: EX/DM to ID, EX/DM to EX.**
- All **write back** executions targeting to \$0~\$31 **are done during the first half of the cycle** in the WB stage.
- All **reads to registers are done during the second half of the cycle** in the ID stage.
- PC is updated in each cycle **after** executing all instructions in each pipeline stage.
- If inserting “NOPs” is needed to resolve hazards, use *sll \$0, \$0, 0*, i.e. the bit stream

10420 CS410001 – Computer Architecture 2016

0x00000000_h. **In your implementation, you should decode** the instruction bit stream 0x00000000_h **as NOP**.

b. Other Constraints

- i. The pipeline is **initialized** with **NOPs in all stages**.
- ii. The simulation of the pipelined processor **terminates** after the **five “halt”** instructions arriving **all the stage**.
- iii. **Register 0** is a **hard-wired 0**; any attempt to write to register 0 takes no effect.
- iv. The instruction memory is of 1K bytes size, the data memory of 1K bytes size.
- v. The executable should be named **pipeline**.
- vi. **To avoid re-execution of some stages, the order of simulating the pipeline is WB → DM → EX → ID → IF.**

3. Input Test Case File and Format

Same as Project 1. Please refer to the specification of Project 1 and *Appendix B*, “*Sample Input*.”

- **Note:** Your test case should cover at least one control hazard and one data hazard resolved by inserting NOPs, and one data hazard cleared by forwarding.

4. Output Requirement

- For each test case, generate the following two output files:
 - a. **snapshot.rpt** : Record all the register values at each cycle.
 - b. **error_dump.rpt** : Record any error messages.
- Place the output files at the same directory where your executable file resides.
- For details, please refer to *Appendix C-2*, “*Sample Output for Project 2*.” and *Appendix D*, “*Error Detection Sample*”.

5. The 2-submission Process

Same as Project 1, but with the following modification: The cloned repository from GitHub should be named studentID_02/. At submission, compress the folder studentID_02 as studentID_02.tar.gz, and upload studentID_02.tar.gz and studentID_report.pdf to the iLMS system.

6. Grading Policy

Same as project 1.

Note: Demo parts will focus on:

- a. Structure of Pipeline
- b. Design of Pipeline
- c. Hazard Concepts
- d. Forwarding Path’s Effect

10420 CS410001 – Computer Architecture 2016

- **Etiquette**
 - a. **Do not plagiarize others' works, or you will fail this course.**
 - b. **No acceptance of late homework.**
 - c. **For details of submission, please note the announcement on the course website.**