**Computer Architecture**

**Project Report**

**Subject** Computer Architecture

**Professor** 김효수 교수님

**Name** 양인수 (20140313)

**Table of Contents**

**1. Brief Project Description**

**2. Source Code Analysis**

**3. Test Results**

**1. Brief Project Description**

The following project is for the class “Computer Architecture”, and is specifically focused on implementing the MIPS pipelined processors with data hazard handling.

**Project Environment**

1. Implementation language : java

텍스트이(가) 표시된 사진

자동 생성된 설명

1. OS: macOS Catalina

텍스트이(가) 표시된 사진

자동 생성된 설명

Please verify that this project cannot be executed by the command:

dhh\_simulator <input\_file\_name>,

since this project was developed in the macOS (does not support .exe)

instead, there is a jar file created and can be ran by the command:

java -jar dhh\_simulator.jar <input\_file\_name>

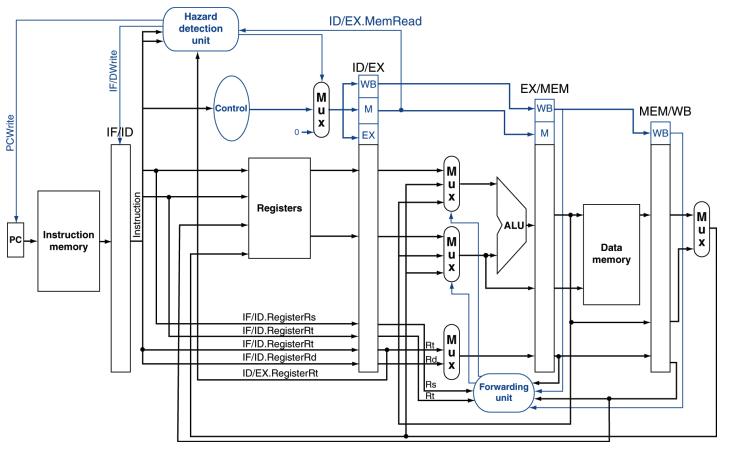
(Details on how to run the project is described in readme.txt)

**2. Source Code Analysis**

There are total 5 .java source code files:

1. Processor.java
2. TextFileReader.java
3. InstructionHandler.java
4. Registers.java
5. DataHazardHandler.java
6. OutputWriter.java

To explain briefly, the 5 classes operate with each other to simulate a MIPS pipelined processor with data hazard handling both EX, MEM Data Hazard and Load-use data hazard.



The Processor.java class contains the main function, and functions as a central unit that processes what the PC, instruction memory, and the forwarding unit would do.

TextFileReader.java is a class that reads in from the input file, and stores the list of instructions in a ArrayList in the Processor.java class named instructions.

InstructionHandler.java is a class that simulates the ID stage, where it translates the instructions to logical numbers, allowing for further arithmetic operations.

Registers.java contain all the registers in the MIPS processor.

DataHazardHandler.java simulates what a Hazard Detection unit would do in the real processor. It detects potential data hazards, then process them for proper pipelining.

OutputWriter.java writes the output as a csv file.

**[Code Analysis : Refer to Processor.java]**

The main function is where the CPU starts.

First, the input file is read and mips instructions are saved in the ArrayList instructions.



After reading input file, the processor immediately starts the counter clock, then the first instruction is carried out to the IF stage. The function stageInstructions works like a dispenser that takes the next instruction to its proper stage. Since this is the first IF stage, the first instruction is carried to the IF stage without any conditions.

텍스트이(가) 표시된 사진

자동 생성된 설명

Then, the processor starts processing the instructions until all the instructions have passed the WB stage. isEnd() function determines if the processor is ready stop handling instructions.



텍스트이(가) 표시된 사진

자동 생성된 설명

The next part is basically a simulation of the IF Stage. It checks if there is any need for stalls, then stages each instructions.

텍스트이(가) 표시된 사진

자동 생성된 설명

Then, all the registers are written in the order of :

MEM/WB -> EX/MEM -> ID/EX -> IF/ID

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

At each stage, it checks for propagation from the previous cycle.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

Then, the processor checks for potential Data Hazards., Then finally one clock cycle ends, then the next clock cycle begins that repeats all the previous steps if the isEnd() is not satisfied.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

More detailed descriptions are written in each line of the source code. Please refer to the comments in the source code.

**3. Test Results**

1. Testing for double data hazard

test1.txt



This input should return the following table:

테이블이(가) 표시된 사진

자동 생성된 설명

The following chart is the output of program when given test1.txt as input.

테이블이(가) 표시된 사진

자동 생성된 설명 텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명



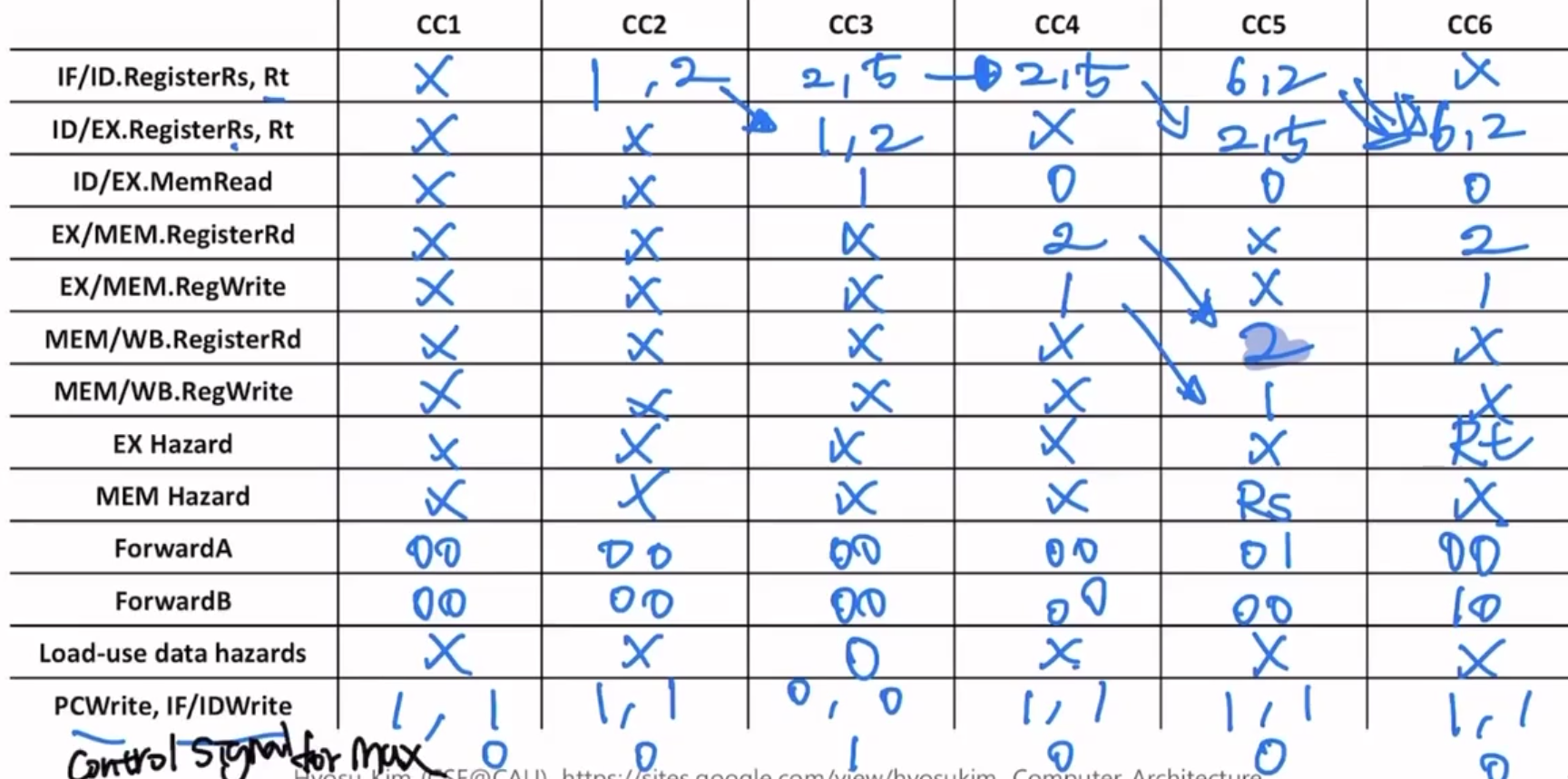
The values prove to be the same. Double data hazard is detected.

The picture on the right is a screenshot of the console screen. For better understanding, the stages of instruction and Register values for each clock cycle are printed.

1. Testing for load-use data hazard & EX Hazard for Rt & MEM Hazard for Rs



This input should return the following table:



The following chart is the output of program when given test2.txt as input.

테이블이(가) 표시된 사진

자동 생성된 설명 텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명



The values prove to be the same. load-use data hazard & EX Hazard for Rt & MEM Hazard for Rs

1. Testing test.in condition



<input table> <output table>

테이블이(가) 표시된 사진

자동 생성된 설명 테이블이(가) 표시된 사진

자동 생성된 설명

All the hazards are detected and no errors occur, however in the project’s processor, the Rd registers are samely written as the Rt values when the instruction is I- format.

텍스트이(가) 표시된 사진

자동 생성된 설명

