Computer Architecture Laboratory
Assignment 3

Develop a single cycle processor simulator for *ToyRISC*.

Input to the Program

1. full path to the configuration file.
   A sample file is given at `src/configuration/config.xml`. This describes the processor to be simulated. This is not of much use in this assignment, but will be utilized in coming assignments.

2. full path to the statistics file to be created. This file contains the statistics of the simulation run – number of instructions executed, number of cycles taken, etc.

3. full path to the object file whose execution is to be simulated.

Output of the Program

- the program must create the statistics file at the specified location.

Broad Outline of the Steps

- **Loading of the program**: Begin with implementing the `loadProgram` function in the file `Simulator.java`. Store the global data and instructions in the approporiate locations in the `mainMemory` structure (see Section "Address Space Layout" in assignment 1). Set the `PC` register to the memory address of the first instruction (remember from assignment 2 that this is the first integer in the object file). Set `x0`= 0, `x1`= $2^{16} - 1$, and `x2`= $2^{16} - 1$.

  You can check the contents of the memory using `processor.getMainMemory()` `.getContentsAsString(<starting-address>, <ending-address>)`.

- **Simulation of the program**:
  - The function `simulate()` in the file `src/generic/Simulator.java` describes the overall loop that captures the simulation. The five stages of the processor are called in-order.
  - As discussed in class, between every two stages, there is a latch. The contents of the different latches are obviously different. Therefore, for each latch type, a separate class has been created. For example, the IF-OF latch is described by the file `src/processor/pipeline/IF_OF_LatchType.java`. An object of this type is created, and references to the same object are given to both the IF stage and the OF stage.

    You have to decide the contents of each latch. `IF_OF_LatchType` has been done for you. Similarly, implement the remaining latches.
  - Implement each of the stages. The IF stage has been implemented for you. Similarly, implement the remaining stages.
  - The simulation ends when an `end` instruction passes through the WB stage. Call `setSimulationComplete()` (defined in the file `src/generic/Simulator.java`) at this point.
  - Set the statistics – number of instructions executed and the number of cycles taken – once the simulation is completed. You may do this at the end of the `simulate()` function. To do the setting, call the appropriate functions in the file `src/generic/Statistics.java`.

    You are encouraged to add statistics of your own in the file `Statistics.java`.
- Tabulate cycles and instruction counts for all programs that you submitted as part of assignment 1, in your report. These programs will function as benchmarks for the our processor designs.

Testing:

- Five example ToyRISC programs, their object codes, and the corresponding expected system state at the end of execution, are given in the supporting files archive.
- Also, the simulator prints a hash value – a hash of the processor state – at the end of the execution. The expected correct hash values are also given.

- Run the script `python test zip.py <path-to-zip-file>` to check your submission. You are expected to submit a zip archive of the src folder.
- The name of your zip archive should be `rollno_lab3.zip eg:cs24bt0xx_lab3.zip`

Note

- All subsequent assignments in this course will involve modifying this set of source files. You will be adding many more files to this project. Please use version controlling extensively, and back up your code frequently.
- You will have **two weeks** to submit this assignment.