README

CS-204 project, RISC-V SIMULATOR (Phase 3):-
Languages:- Python

TEAM MEMBERS(Contributions):-
Ajey Singh - 2018eeb1133

1. Worked on detection of all possible data hazards (in case of pipelining with data forwarding paths) and the manner in which each one of them can be tackled.
2. Inclusion of additional hardware in the data path to resolve for some data hazards to reduce stalls.
3. Contribution in developing pipeline registers.

Yogesh Vaidhya - 2018eeb1277

1. Worked on making pipeline execution of instruction, handling stalls, flushing logic.
2. Worked on control hazard detection, branch prediction, data hazard detection with and without forwarding.
3. Contribution in modifying pipeline registers.

Garima Agarawal - 2018eeb1146

1. Contribution in making pipelining and working on data hazard detection .
2. Contribution in debugging of the code.

Ish Rajesh Shelley - 2018eeb1153

1. Major part of my contribution was in debugging.

HOW TO RUN:-
Right Click On Folder -> Open in terminal
On the terminal -

```
python src/<file_name.py>
```

File contain 3 files python files
1. Phase3_nonpipeline.py – Use this file for non-pipeline implementation
2. Phase3_only_stalling.py – Use this file for pipeline without forwarding
3. Phase3_stalling_forwarding – Use this file for pipeline with forwarding

FEATURES:-

1.1. Functionality to switch between
    1.1.1. Non- pipelined
    1.1.2. Pipelined with stalls
    1.1.3. Pipelined with Data forwarding

2. Functionality to output register values and values of the interstate buffer
3. Functionality to run cycle by cycle

4. Present the number of branch mispredictions, number of stalls due to control and data hazards etc.
5. Calculates CPI of the program and displays total number of control, data transfer and ALU instructions.
6. Print register values and inter state buffers after every cycle.
7. All possible data hazards have been taken care of by inclusion of additional hardware such as muxRM and muxRZ to reduce the number of stalls and thereby, taking CPI as close to 1 as possible.

ASSUMPTIONS:-

“.mc” file generated as output by phase 1 would be free of syntax errors and would not contain ld and sd instructions.

NOTE:-

Since after a single run of source code, the memory is being written in the .mc file. So, please ensure that before running the file again, .mc file is cleared of the memory printed to it in the first run otherwise it won't be able to read .mc file correctly.