# Lab5 Pipeline Stall Detector/Simulator

**CO21BTECH11004**

## Approach: -

Input: -

- Read the input file line by line
- Split the input, meaning get Inst, register names, and offset (for load/store) and store them as vector<string> for each instruction.
- Also, run checks for the correct register name in the input and R-type, I-type, load, and store instruction names.

## Part1 : -

- Iterate through each instruction.
- If R-Type, I-type or load instructions, call addNop function.
- addNop function do the following: -
    - Stores current destination register in the variable ‘r’.
    - If r == x1 or r == “zero”,  do nothing.
    - If ‘r’ matches the next instruction’s source register, add two nops after the current instruction.
    - Else if ‘r’ matches the 2nd next instruction’s source register, add one nop before the 2nd next instruction.

## Part2: -

- Iterate through each instruction.
- If load instruction, call addNop function.
- addNop function do the following: -
    - Stores current destination register.
    - If r == x1 or r == “zero”,  do nothing.
    - If ‘r’ matches the next instruction’s source register, add one nop after the current instruction.

## Output:  -

- Iterate through the updated Instructions vector (containing nops).

- If nop is there, print nop.
- Else, print the corresponding instruction stored in the input vector, which contains the instructions in the original form we got from the input file.
- Print total cycles = size of updates instruction vector + 4.

Two code files: -

- part1.cpp: - assuming no data forwarding and no hazard detection.
- part2.cpp: - assuming data forwarding without hazard detection.

Testing the code: -

- Made test cases manually and used back exercises in the book.
- Compared to the output.