Lab-5 Report
Hazard Detection And Forwarding

Name:Aditya Garg
Roll Number:CS22BTECH11002

November 6, 2023

1 Coding Approach

- Before Coding the problem , I wrote various cases possible where we should place "nop" for the proper function of any given code .
- The 3 different checks are required to be implemented - load instruction , store instruction and all the rest (R/I since branch not present) instructions. For a given assembly instruction , only 2 instructions above it are required to be checked for adding nop in both the cases .

  Without Forwarding

  – In general , if "rd" of previous instruction is used as "rs1" or "rs2" of current instruction an addition of 2 nop would resolve the issue.
  – A single "nop" would be required in the above case if there is an instruction between them acting as an "nop"

  With Forwarding

  – The only case where we need to add nop is when the rd of a load instruction is an rs1 or rs2 of an immediate next instruction .
- I created a 2-D vector of type strings , which stored the parsed input . Parsing was done based on the "," in the instruction and rd , rs1 , rs2 , offset and number of registers in the instruction is stored .
- In order to ensure the code gives correct output for mix of register and ABI name , I created a map , mapping each of the register to its ABI name and compared the map value for equality .
- I created 2 vectors of pair int , bool with size as the number of lines in the input . The vector were initialised to false and after iterating over the

input stored the number of nop required ,true on the index equal to the line number where data hazard was required.

- While iterating , based on the parsed input , I would identify it as either a load , store or R/I type instruction . Compare with the map value of rd of previous 2 instruction and mark the position of data hazard as true. In the case when forwarding was implemented only in the load case , position was marked true .
- If there were data hazards in the both the previous instruction , only max of the nop was added to minimise the number of nop required.
- Another 2 variables initialised to number of lines in the input , stored the total number of cycles required for the correct execution . The value of the variable incremented by the amount of the nop added .
- Finally I outputted the modified code . check if the given line is marked true , if it is add the required amount of nop stored in the vector itself and continue the procedure .

2 Testing Approach

- For testing , first I checked the basic detection between two consecutive R type having the subsequent rs1/rs2 and rd same , Combination of load and R type.
- In order to check all possibilities , I took combination of all scenarios (load with R , load with load , load with store etc ..) and generated the test cases .
- I tried mixing ABI name and the register to find if any mistake in the map . All the test cases are attached in the submission.
- Multiple test cases have been added in TestCases.txt file along with the their output for verification.