CODING APPROACH

The code has been written in c++.

One main observation was that only the rd of an instruction can cause requirement of nops to stall the program. Therefore, a count is kept(in register_update) of how long an register can cause nops to reach. This is updated by 2 in case of ld, lui and R/I type commands, when there is no forwarding, and 1 if it is ld and there is forwarding. These values are decremented by 1 every instruction, and incase it is non zero, we add the nops to be added into a nops[] array, and set this counter to 0.

There is also some special case to consider, like when rd is 0, in which case we do not update the vector. Also a map has been used to map registers to their register values.

The instructions are extracted by simply extracting from the front and making cases. If it is null it wont affect the cases anyway.

Finally the cycles are calculated simply by counting the number of instructions including nops, and adding 4.

To check if the code was correct, I tried using all the R, S, I type commands. I also made some special cases(like when a unrelated instruction can reduce nops by 1) and also check whether the aliases worked.